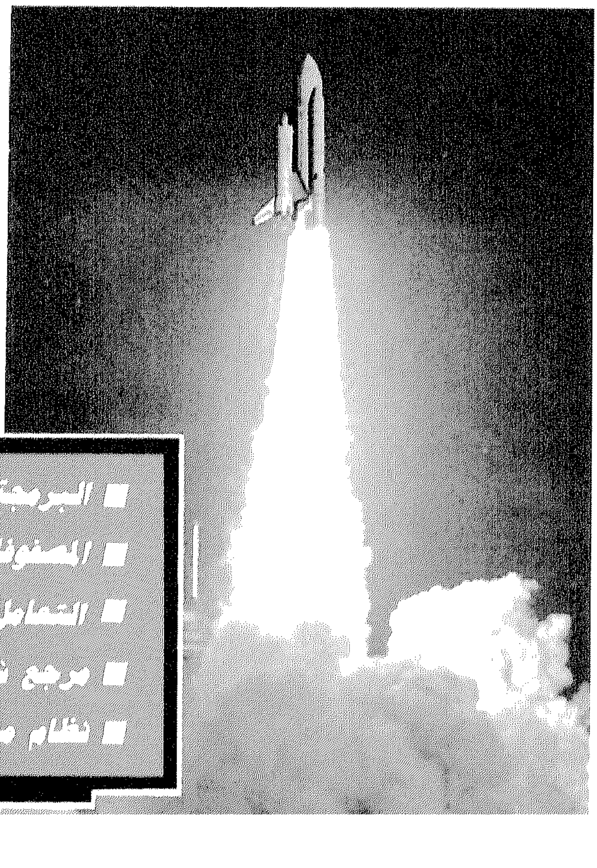
مجدي محمد أبو العطا

المرجع الأساسي

لقاعدة البيانات

Clipper™



- البرمجة باستخدام «كلبر»
- المصفوفات . شبكات الاتصالات
- التعامل مع أخطاء البرامج
- مرجع شامل للأوامر والوظائف
- نظام متكامل للمبيعات

الجزء الثاني

الطبعة الأولى
١٤١١هـ - ١٩٩١م

اهداءات 2002

المهندس/ سيد مصطفى أبو السعود

القاهرة

الباب الرابع

مرجع الأوامر والوظائف

الفصل الثالث عشر

مرجع الأوامر

تتعامل قاعدة البيانات cLipper شأنها شأن أي لغة برمجة مع الأوامر التي يفهمها المترجم الخاص بها والمعرفة سلفاً بواسطة مصممي لغة البرمجة. ويشتمل هذا الفصل على جميع الأوامر التي تتعامل معه «كلبر» مرتبة حسب الترتيب الأبجدي للحروف الانجليزية لسهولة الوصول إلى أي أمر بسرعة وعند شرح كل أمر على حدة وضعنا بين يديك كل المعلومات الضرورية عن هذا الأمر مثل:

يتكون أي أمر من أوامر قاعدة البيانات Clipper على الأقل من كلمة واحدة تسمى الفعل (Verb أو Keyword) ويجوز أن يشتمل بالإضافة إلى الفعل على العديد من الاختيارات بعد ذلك وتأخذ أوامر قاعدة البيانات cLipper الشكل العام الآتي:

VERB [<*scope*>] [<*expression list*>] [FOR/WHILE <*condition*>]

[OFF] [TO PRINT] [TO FILE <*filename*>]

وتلاحظ من الشكل العام للأوامر أننا كتبنا الكلمات الثابتة في الأمر بالحروف الكبيرة (Upper Case) . أما الكلمات التي ستستبدل بقيم من خارج الأمر فكتبناها بالحروف الصغيرة (Lower Case) .

مثال:

LIST	NEXT 5	STUDENTNO,LASTNAME	FOR LASTNAME = "ABU AL-ATA"
↓	↓	↓	↓
VERB	<*Scope*>	<*expression list*>	<*condition*>
الأمر	المدى	الحقول	السجلات التي تخص

ويمكن أن يستخدم هذا الأمر بإحدى الصيغ الآتية:

```
LIST
LIST  NEXT 5
LIST  NEXT  5  STUDENTNO,LASTNAME
LIST  NEXT  5  STUDENTNO,LASTNAME  FOR  LASTNAME = "Abu AL-Ata"
LIST  NEXT 5  STUDENTNO,LASTNAME  FOR  LASTNAME = "ABU AL-ATA"  TO PRINT
```

الأمر ?/??

يظهر عبارة أو حقل أو أكثر.

الشكل العام:

? *<expression list>*

?? *<expression list>*

حيث:

<expression list> : يمكن أن تكون عبارة حرفية أو رقمية أو منطقية أو تاريخية.

الشرح:

يظهر الأمر ? محتويات عبارة أو عبارات قد تكون رقيمة أو حرفية أو تاريخية أو منطقية ابتداء من السطر التالي لمكان المؤشر على الشاشة.

أما الأمر ?? فيعمل نفس العمل إلا أنه يظهر محتويات العبارة أو العبارات المذكورة حيث يقف المؤشر على الشاشة.

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثال:

المثال الآتي يظهر سطرين متتابعين:

```
? "This is a numeric expression"        && تعبير حرفي
? (10+10)*50*((20-10)/2))               && تعبير رقمي
```

النتيجة

```
This is a numeric expression
      5000
```

بينما يظهر المثال النتيجة على نفس السطر

```
USE STUDENTS
? "Student's name: "                 && (expC) تعبير حرفي
?? TRIM(FIRSTNAME)+LASTNAME           && تعبير حرفي يستخدم محتويات حقل
```

النتيجة

```
Student's name: HKALID BIN NASER
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة :

@...SAY – TEXT...ENDTEXT – DISPLAY – SET PRINT

الأمـر @...BOX

يظهر برواز على الشاشة.

الشكل العام:

@ <expN1>, <expN2>, <expN3>, <expN4> BOX <expC>

حيــث:

<expN1>	:	رقم يشير إلى مكان السطر العلوي من المستطيل على الشاشة.
<expN2>	:	رقم يشير إلى العامود الذي سيبدأ عنده المبتطيل من جهة اليسار.
<expN3>	:	رقم يشير إلى مكان السطر السفلي من المستطيل على الشاشة.
<expN4>	:	رقم يشير إلى العامود الذي سينتهي عنده المستطيل من جهة اليمين.
<expC>	:	تعبير حرفي يتكون من مجموعة حروف تصل إلى ثمانية وهي الحروف التي ستستخدم في رسم الاطار أو البرواز بالاضافة إلى حرف تاسع يستخدم لملء الفراغ الموجود داخل المستطيل.

الشـــرح:

يتم تحديد مكان الزوايا الأربع من البرواز بالاختيارات

expN1, expN2, expN3, expN4

أما الحروف أو العلامات التي ستستخدم لرسم البرواز فيحددها الاختيار expC ويمكن أن يشتمل حتى ٩ رموز أو حروف، واحدة لكل ركن وواحدة لكل ضلع والتاسعة لخلفية البرواز، وتخصص قاعدة البيانات الحرف الأول للركن اليسار العلوي والثاني للسطر العلوي والثالث للركن اليمين العلوي والرابع للضلع الأيمن والخامس للركن اليمين السفلي والسادس للسطر السفلي والسابع للركن اليسار السفلي والثامن للضلع الأيسر.

وننصح بوضع الحروف الثمانية أو التسعة داخل حقل ذاكرة بدلا من رصها في الأمر نفسه هكذا

SINGLE = CHR(218) + CHR(196) + CHR(191) + CHR(179) +;
CHR(217) + CHR(196) + CHR(192) + CHR(179)

ثم استخدم SINGLE بدلاً من <*expC*> في الأمر الرسم برواز بخط فردي .

الاختلاف عن *dBASE III PLUS* : غير موجود بها .

أمثلـــة :

مثال ١ : لرسم إطار ابتداءً من السطر العاشر والعامود العاشر وانتهاءً بالسطر الخامس عشر والعامود الثلاثين بحيث تستخدم الأرقام من ١ إلى ٨ في رسم الاطار. ومن المثـال تستطيع تحديد اتجاه رسم البرواز وهو في اتجاه عقارب الساعة استخدم هذا الأمر:

@ 10, 10, 15, 30 BOX "1 2 3 4 5 6 7 8"

```
SINGLE = CHR(218)+CHR(196)+CHR(191)+CHR(179)+CHR(217) ;
      +CHR(196)+CHR(192)+CHR(197)
```

مثال ٢ : المثال التالي يستخدم لرسم إطار يستخدم خطا فرديا في جميع الزوايا والأضلاع .

```
SINGLE = CHR(218)+CHR(196)+CHR(191)+CHR(179)+CHR(217) ;
      +CHR(196)+CHR(192)+CHR(197)
@ 10,10,15,30 BOX SINGLE
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

@...CLEAR - @...TO

الأمر @...CLEAR

يحذف منطقة محددة من الشاشة.

الشكل العام:

@ <*expN1*>, <*expN2*>, CLEAR [TO <*expN3*>, <*expN4*>

حيث:

<*expN1*>	:	رقم السطر الذي سيبدأ الحذف عنده (تنظيف الشاشة).
<*expN2*>	:	رقم العامود الذي سيبدأ الحذف عنده (تنظيف الشاشة).
<*expN3*>	:	رقم السطر الذي ستنتهي عنده منطقة تنظيف الشاشة.
<*expN4*>	:	رقم العامود الذي سينتهي عنده منطقة تنظيف الشاشة.

الشرح:

إذا لم يحدد الاختيار TO فإن تنظيف الشاشة سيستمر ابتداء من السطر والعمود المحددين حتى نهايتها. ويمكن الاستفادة من هذا الأمر بصفة خاصة عند التعامل مع القوائم الرأسية بالطريقة التالية:

استخدم أولا أمر SAVE SCREEN لحفظ الشاشة كما هي ثم استخدم أمر @...CLEAR لتنظيف جزء من الشاشة لاظهار نافذة محله وعند الانتهاء من الحاجة إلى النافذة استخدم أمر RESTORE SCREEN لاسترجاع الشاشة التي حفظتها.

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثال:

لحذف المنطقة التي تبدأ من السطر العاشر والعمود العاشر وتنتهي بالسطر الرابع عشر والعمود السبعين

@ 10, 10 CLEAR TO 14, 70

ولحذف المنطقة التي تبدأ من السطر العاشر والعمود العاشر إلى نهاية الشاشة

@ 10, 10 CLEAR

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

@...BOX – CLEAR – SCROLL( )

الأمر @...PROMPT

يظهر رسالة محث تستخدم كأحد اختيارات القائمة الرأسية.

الشكل العام:

@ <*expN1*>, <*expN2*> PROMPT <*expC*> [MESSAGE <*expC*>]

حيث:

<*expN1*>	:	رقم السطر الذي سيظهر عنده الاختيار داخل القائمة.
<*expN2*>	:	رقم العامود الذي سيظهر عنده الاختيار داخل القائمة.
<*expC*>	:	الرسالة التي ستظهر عند السطر والعمود المحددين.
MESSAGE <*expC*>	:	يمكن اختيارها لاظهار رسالة إضافية في المكان الذي يحدده أمر SET MESSAGE.

الشـــرح:

يستخدم هذا الأمر لاظهار اختيار/ اختيارات القوائم الرأسية (Light – bar menu) ويمكن تعريف رسالة لتشرح هذا الاختيار إذا أضيف إليه الاختيار Message. ويمكن تحديد حتى ٣٢ اختيار (PROMPT) للقائمة الواحدة. ويتم الانتقال بين الاختيارات باستخدام مفاتيح الأسهم. استخدم أمر MENU TO للانتقال بين الاختيارات بالترتيب الذي وردت به في البرنامج. واستخدم أمر SET WRAP ON لتتمكن من الانتقال إلى أول اختيار إذا وصلت الأخير.

الاختلاف عن *dBASE III PLUS* : غير موجود بها.

مثـــال :

المثال التالي يظهر قائمة رأسية تتكون من أربعة اختيارات ويظهر مع كل اختيار رسالة في منتصف سطر ٢٤ تشرح معناه

```
CLEAR                              && نظف الشاشة
SET WRAP ON                        && دع المؤشر يتحرك من اسفل لاعلى
SET MESSAGE TO 24 CENTER
@ 10,5 PROMPT "Add item          " MESSAGE "Add a new item"
@ 11,5 PROMPT "Edit item         " MESSAGE "Edit an existing item"
@ 12,5 PROMPT "Delete item       " MESSAGE "Delete item"
@ 13,5 PROMPT "Quit              " MESSAGE "Quit to DOS"
MENU TO CHOICE                     && اسمح بقبول الاختيارات
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

MENU TO – SET COLOR – SET MESSAGE – SET WRAP – ACHOICE( )

الأمر @...SAY...GET

يظهر البيانات في المكان المحدد في الأمر.

الشكل العام:

@ <*row*>, <*col*> [[SAY <*expression*> [PICTURE <*template*>]]

[[GET <*variable*> [PICTURE <*template*>]

[RANGE <*lower bound*>, <*upper bound*>] [VALID <*expL*>]]]

حيـــث:

<*row*>, <*col*>	:	<*row*> هي رقم السطر، و <*col*> هي رقم العمود.
<*expression*>	:	قد يكون محتويات حقل أو ذاكرة أو عبارة أو بعض أو كل ذلك.
<*template*>	:	حروف تتحكم في شكل البيانات الخارجة أو الداخلة إلى الحاسب.
<*variables*>	:	قد تكون محتويات حقل في الملف أو محتويات ذاكرة.
<*lower*>, <*upper*>	:	قد يكون رقما أو تاريخا.
VALID	:	لقبول المدخلات في حالة GET إذا تطابقت مع شرط معين.
<*expL*>	:	الشرط الذي ستقيّم العبارة الداخلة بناء عليه.

الشـــرح:

يشتمل هذا الأمر على اختيارين أساسيين: **الأول SAY ـ والثاني GET**، بالاضافة إلى اختيارات أخرى.

وكل من هذين الاختيارين يشتمل على مجموعة اختيارات أخرى ويسمح الاختيار SAY بإظهار محتويات حقول الملفات أو الذاكرة <*expression*> في السطر والعمود المحددين في الأمر (<*row*>, <*col*>) ويمكن التحكم في شكل هذه البيانات

إذا اشتمل على الاختيار PICTURE .

أمـا الاختيار GET فيستخـدم لاظهـار بيانات موجودة بالملف أو الذاكرة مع إمكانية التعديل في هذه البيانات وذلك في السطر والعمود المحددين. ويمكن التحكم أيضا في شكل البيانات الداخلة إلى الحاسـب إذا اشتمـل على أحـد الاختيارين PIC-TURE أو RANGE أو كليهما. والاختيار GET خاص بإظهار وتعديل البيانات على الشاشة فقط فإذا اختيرت الطابعة بالأمر SET DEVICE TO PRINT فستهمل قاعدة البيانات هذا الأمر.

ولذلك فمن المناسب أن تستخدم SAY للبيانات التي تريد إظهارها فقط بدون صلاحيات التعديل فيها و GET للبيانات التي ترغب في تعديلها.

وإليك المفردات الواردة بهذا الأمر والاختيارات المتاحة لكل منها:

الاختيـار *<row>*, *<col>* :

يمثل مكان إظهار المحتويات على الشاشة في السطر والعمود المحددين وترقم السطور على الشاشة من صفر إلى ٢٤ (من أعلى إلى أسفل الشاشة) والأعمدة من صفر إلى ٧٩ (من يسار إلى يمين الشاشة).

الاختيـار PICTURE :

يسمح بإظهار المخرجات بشكل مغاير لشكلها داخل الحقل أو الذاكرة. ويعطي فرصة للتحكم في البيانات الداخلة إلى الحقول أو الذاكرة. ويشتمل على واحد أو أكثر من الرموز الموضحة في الجدول ١ ـ ١ ولكل رمز دلالة معينة (حسب إرشادات الجدول) يطلق عليها Function .

الوظائف Functions :

الرمز	معنـــــــــاه
A	تسمح بحروف أبجدية فقط .
B	تضع بداية واحدة للبيانات الرقمية من ناحية الشمال .
C	تظهر الرمزين CR بعد الأرقام الموجبة فقط وتستخدم مع الاختيار SAY فقط .
D	تعامل الحقل على أنه تاريخ Date عند إدخال بياناته .
E	تعامل الحقل على أنه تاريخ بالشكل الأوروبي .
R	تظهر سلسلة حروف بدون تخزينها إلى الحقل .
S <*n*>	تحدد أقصى اتساع للحقل المظهر بالرقم الموجود في <*n*> وتعطي إمكانية إزاحة البيانات داخل الحدود المسموح بها لرؤية باقي محتويات الحقل .
X	تظهر الرمزين DB أمام الأرقام السالبة فقط وتستخدم مع الاختيار SAY
Z	تظهر محتويات الحقول الرقمية كفراغات إذا كانت تشتمل على القيمة صفر.
(	يظهر الأرقام السالبة بين قوسين وكذلك الفراغات إن وجدت .
)	يظهر الأرقام السالبة بين قوسين بدون الفراغات الموجودة .
!	تحـول الحـروف الأبـجـديـة الانـجـليزيـة إلى الحـروف الـكـبـيرة (Uppercase) .

الجـدول ١ ـ ١

وقد يشتمل بالاضافة إلى ذلك على سلسلة رموز يقابل كل رمز منها حرف عند إدخال البيانات أو إظهارها. ويسمى Template . وهذه المجموعة من الرموز موضحة

في الجدول ٢ ـ ١ ويجب أن توضع هذه الرموز بين علامتي التنصيص "" أو ''

فإذا كنت ستستخدم أحد الرموز الموضحة بالجدول رقم ١ ـ ١ (Functions) فيجب أن تسبق علامة @ هذه الرموز. وإذا كنت تريد إضافة أحد رموز الجدول رقم ٢ ـ ١ (Template) فيجب أن تضع مسافة خالية بين النوعين. بمعنى أن الترتيب داخل الاختيار PICTURE يجب أن يتم كالآتي:

PICTURE *"@ <function> <space> <template>"*

Template

الرمز	معنـــــاه
A	يسمح بإدخال حروف أبجدية فقط.
L	يسمح بإدخال بيانات منطقية. أو بإظهارها في صورة .T. أو .F.
N	يسمح بإدخال حروف وأرقام فقط.
X	يسمح بإدخال جميع الحروف.
Y	يسمح بإدخال أحد الحروف Nn Yy في الحقول المنطقية.
9	يسمح بإدخال أرقام فقط للبيانات الحرفية وبإدخال الأرقام والاشارات للبيانات الرقمية.
#	تسمح بإدخال فراغات وأرقام وإشارات فقط.
!	تحول الحروف الصغيرة إلى كبيرة.
$	تظهر علامة الدولار الأمريكي أمام البيانات الرقمية.
*	تظهر علامة * أمام البيانات الرقمية.
.	تحدد مكان العلاقة العشرية.
,	تستخدم كفاصل بين الأرقام الموجودة على شمال العلامة العشرية.

الجدول ٢ ـ ١

انظر هذا المثال

PICTURE "@x ***,*** **"

في هذا الجزء من الأمر فإن علامتي التنصيص ضرورية ومنه تلاحظ الآتي:

- استخدمنا علامة @ لأن الاختيار اشتمل على أحد رموز الجدول ١ ـ ١ (Function) وهو X ويعني إظهار حرفي DB أمام الأرقام السالبة.
- المسافة الخالية لتفصل بين المجموعتين، (Function & Template).
- مجموعة علامات * تسمى Template وتستخدم كل علامة للتعويض عن غياب رقم. بمعنى أن هذه العلامة يجب أن تحل محل الأصفار التي تظهر على شمال الرقم وذلك بنفس عدد علامات *

الاختيار RANGE :

يستخدم هذا الاختيار لتحديد حد أقصى وحد أدنى للرقم أو التاريخ الذي سيدخل إلى الحاسب بمعنى أن أي رقم أو تاريخ لا يقع بين الحد الأعلى والحد الأدنى لن يدخل إلى الحاسب. وستظهر رسالة للمستخدم تشتمل على الحدود العليا والدنيا المقبولة وتطلب منه ضغط مسطرة المسافات لاعادة المحاولة.

ويجب أن تكون القيمة المحددة في الأمر إما قيمة رقمية أو تاريخية بناء على الحقل الذي ستدخل إليه ولكي يفهم الحاسب أن القيمة الداخلة إليه تعامل معاملة التاريخ فيجب أن تستخدم الوظيفة ( )DTOC فمثلا الاختيار:

RANGE CTOD ('01/01/90'), CTOD ('12/31/90')

لن يقبل تاريخا يقع خارج عام ١٩٩٠.

ويمكن أن يشتمل الاختيار RANGE على الحد الأعلى فقط هكذا:

RANGE , CTOD ('12/31/90')

أو على الحد الأدنى فقط هكذا:

RANGE CTOD ('01/01/90'),

أما إذا اهمل هذا الاختيار فمعنى ذلك عدم تقييد المدخلات بحد أعلى أو أدنى.

الاختيـار *VALID* :

يستخدم لتصحيح المدخلات إلى الحقول التي تدخل بياناتها بالاختيار GET وذلـك باتبـاع الاختيار بتعبـير منـطقي . ويتم تقييم هذا التعبير فإذا كانت النتيجة صحيحة (.T.) قلبت المدخلات إلى الحقل، وإلا فلن تستطيع الانتقال من الحقل إلا إذا ضغـطت مفتاح Esc ، ويمكن أن يكون التعبير المنطقي تعبيرا داخل الأمر، كما يمكن أن يكون وظيفة خاصة (UDF) يتم استدعائها لتقوم بعملية التقييم .

ملاحظات هامة :

- أمر READ يسمح بإظهار مجموعة أوامر GET...@السابقة له مع إمكانية التعديل فيها . ولذلك إذا كنت تريد أن تدخل بياناتك على أكثر من صفحة فيجب أن تضع أمر READبعد آخر أمر في مجموعة أوامر GET...@ التي تريد أن تظهر محتوياتها على الشاشة كصفحة واحدة. ثم تضع أمر READ مرة أخرى عند نهاية المجموعة التالية .
- استخدام أمر CLEAR GETSبدلاً من أمر READ ، يظهر الشاشة بالشكل الذي تظهر به في حالة إدخال أو تعديل البيانات إلا أنها لن تقبل منك إدخال بيانات إليها . ويمكنك الاستفادة من ذلك في حالة إظهار الشاشة بنفس الشكل الذي تظهر به في حالة الاضافة أو التعديل بدون صلاحيات التعديل فيها كما يحدث في حالة إظهارها كمعلومات فقط .
- يمكن تصميم شاشة مساعدة للمستفيد من النظام لاظهار معلومات عن النظام باستخـدام مجمـوعة أوامر SAY...GET...@ ويتم ذلك بتخصيص أحد مفاتيح الوظائف وليكن F1 لاستدعاء إجراء معين .
- إذا اشتمـل الأمر على الاختيارين SAYو GETفإن قاعدة البيانات تفصل بينهما بمسافة خالية .

الاختلاف عن *dBASE III PLUS* : لا تشتمل «دي بيس» على الاختيار VALID وفيما عدا ذلك فالأمران متطابقان .

أمثلــــة :

(١) في المثال الآتي يضع الأمر الأول ١٤ فراغا في مخزن بالذاكرة اسمه MNAME ويظهـر الأمـر الثـاني عبـارة "Enter your name" ابتداء من السطر الخامس والعمود الخامس عشر متبوعة بالفراغات المخزنة بالأمر الأول ليحل محلها الاسم الذي سيدخل إلى الحاسب ويسمح الاختيار PICTURE بإدخال الاسم بالحروف الكبيرة (Upper Case Letters) فقط .

```
MNAME = SPACE(14)
@ 5,15 SAY "Enter your name:" GET MNAME PICTURE "@!"
READ
```

(٢) يسمح المثال التالي بإدخال بيانات رقميـة تتكون من خانتين اثنتين وتقـل عن الرقـم ٦٠

```
MAGE = 0
@ 6,15 SAY "Enter your age:" GET MAGE PICTURE "99" PANGE  ,60
READ
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

@...CLEAR – @...TO – READ – CLEAR GETS

SET CONFIRM – ROW( ) – COL( ) – PROW( ) – PCOL( )

SET DEVICE – SET DELIMITERS – SET PRC ( ) – TEXT

الأمر @...TO

يرسم خطوط أو مستطيلات .

الشكل العام:

@ <row1>, <col1>, <row2> TO <col2> [DOUBLE]

حيـــث:

<row1>, <col1>	:	بداية السطر أو المستطيل .
<row2>, <col2>	:	نهاية السطر أو المستطيل .
DOUBLE	:	ترسم سطر أو مستطيل مزدوج .

الشـــــرح:

يستخدم هذا الأمر لرسم خط مستقيم أو مستطيل على الشاشة. ويحدد الركن اليسار العلوي من المستطيل بالرقم الموجود في <row1> و<col1> ويحدد الركن اليمين السفلي من المستطيل بقيمة <row2> و<col2> فإذا اشتمل الأمر على الاختيار DOU-BLE فإن الخط أو المستطيل المطلوب يرسم مزدوجا. إذا كانت كل من <row1> و<row2> واحـــدة فإن الخط سيرسم أفقيا. وإذا كانت كل من <col1> و<col2> واحدة فإن الخط سيرسم رأسيا .

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف .

مثـــال:

يرسم الأمر التالي مستطيلا مزدوجا على الشاشة يبدأ من السطر الأول والعمود العاشر وينتهي بالسطر الخامس والعمود السبعين

```
@ 1,10 TO 5,70 DOUBLE
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

@...BOX – @...CLEAR – ACHOICE( ) – DBEDIT( ) – SCROLL( )

الأمر ACCEPT

يخصص عبارة حرفية لحقل ذاكرة (Memory Variale) .

الشكل العام:

ACCEPT [<*prompt*>] TO <*memvarC*>

حيث:

prompt : عبارة حرفية .

TO <*memvarC*> : تحدد اسم حقل الذاكرة الذي ستوضع به المدخلات.

الشرح:

يستخدم هذا الأمر عادة داخل البرنامج لوضع العبارة الحرفية التي تدخل من لوحة المفاتيح في حقل الذاكرة المحدد في الأمر. مع إمكانية إظهار رسالة توجيه (prompt) على الشاشة . يستبدل الاختيار <*prompt*> الموجود بالأمر بالرسالة أو العبارة التي ستظهر على الشاشة. فإذا اشتمل الأمر على هذا الاختيار فإن هذه الرسالة أو العبارة تظهر على الشاشة قبل إدخال أي معلومات من لوحة المفاتيح . وتخزن المعلومات التي تدخل من لوحة المفاتيح في حقل ذاكرة دائما حرفي (Character) . الحد الأقصى لطول الرسالة أو العبارة التي تخزن بالذاكرة هو ٢٥٤ حرفا. وتعرف قاعدة البيانات نهاية الرسالة بضغط مفتاح الادخال Enter . فإذا ضغط المستخدم مفتاح الادخال بدون إدخال أي نص فإن حقل الذاكرة لن يحتوي على شيء.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثال:

يظهر الأمر الآتي رسالة:

"Enter the Item Number"

ويضع القيمة التي تدخل من لوحة المفاتيح في حقل ذاكرة اسمه ITEM_NO .

```
ACCEPT "Enter the item_no: " TO ITEM_NO
```

ولاظهار محتويات ITEM_NO ادخل الأمر الآتي :

```
? ITEM_NO
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

@...SAY...GET - INPUT - WAIT - INKEY( )

الأمر APPEND BLANK

يضيف سجلا خالياً في نهاية الملف المفتوح.

الشكل العام:

APPEND BLANK

الشرح:

يستخدم هذا الأمر لاضافة سجل بدون بيانات في نهاية ملف قاعدة البيانات المفتوح.

الاختلاف عن *dBASE III PLUS*: يستخدم أمر APPEND في dBASE III PLUS لاستدعاء شاشة كاملة يتم من خلالها إضافة سجل إلى الملف. أما في CLIPPER فإن الأمر مقبول فقط بصيغة APPEND BLANK .

مثال:

المثال التالي يستخدم ملف STUDENTS.dbf لاضافة سجل بدون بيانات في نهاية الملف.

```
USE STUDENTS
?LASTREC()                    && 15 إجابة كلبر:
APPEND BLANK
?LASTREC()                    && 16 إجابة كلبر:
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

APPEND FROM

الأمر APPEND FROM

يضيف سجلات من ملف موجود على القرص إلى نهاية الملف المفتوح .

الشكل العام:

APPEND [<*Scope*>] [FIELDS <*list*>] FROM <*file*>/(<*expC*>)

[FOR <*Condition*>] [WHILE <*Condition*>]

SDF/DELIMITED [WITH BLANK/ <*delimiter*>/ (<*expC*>)]]

حيـــث:

<*Scope*>	:	السجلات المختارة للنسخ .
FIELDS	:	الحقول المختارة لنسخها .
<Condition> FOR	:	تستخدم لاضافة السجلات التي تتطابق مع الحالة المشروحة إلى الملف المفتوح .
<*Condition*> WHILE	:	تستخدم لاضافة سجلات من ملف طالما أن الحالة المشروحة في الأمر صحيحة .
SDF	:	ملـف نصـــي .

الشـــرح:

يستخدم هذا الأمر لاضافة سجلات في نهاية الملف المفتوح من ملف آخر موجود على القرص. فإذا لم يشتمل الملف الآخر على اسم ممتد فتفترض قاعدة البيانات أنه (.dbf) ويمكن اختيار سجلات معينة لاضافتها للملف المفتوح إذا أضيف الاختيار <*Scope*> أو الاختيار FOR/WHILE للأمر كما يمكن اختيار حقول معينة إذا أضيف الاختيار [FIELDS <*list*>] للأمر.

يستخدم الاختيار [DELIMITED [WITH BLANK/ <*delimiter*> إذا كانت السجـلات المضافة تشتمل على فاصل بين الحقول مثل المسافة الخالية أو علامتي التنصيص "" .

فإذا كانت السجلات المضافة موجودة على ملف نصي مسجل بشفرة ASCII والحقول مفصولة بعلامة (,) Comma ، والبيانات نفسها محاطة بعلامتي التنصيص "" فيكفي أن نكتب كلمة DELIMITED . وإذا كان الفاصل بين السجلات عبارة عن مسافة خالية فإن .الاختيار يستخدم هكذا

DELIMITED WITH BLANK

وإذا كان الفاصل بين السجلات عبارة عن رمز آخر غير المسافة الخالية مثل الفاصلة (;) أو الشرطة (ـ) فإن الأمر يستخدم هكذا

DELIMITED WITH <*delimiter*>

وتستبدل كلمة delimiter بالحرف أو العلامة المستخدمة كفاصل بين الحقول في الملف المضاف.

يستخدم الاختيار DELIMITED [WITH BLANK/ <*delimiter*>]

إذا كانت السجلات المضافة تشتمل على فاصل بين الحقول مثل المسافة الخالية أو علامتي التنصيص "" .

فإذا كانت السجلات المضافة موجودة على ملف نصي مسجل بشفرة ASCII والحقول مفصولة بعلامة (,) Comma ، والبيانات نفسها محاطة بعلامتي التنصيص "" فيكفي أن نكتب كلمة DELIMITED . وإذا كان الفاصل بين السجلات عبارة عن مسافة خالية فإن الاختيار يستخدم هكذا

DELIMITED WITH BLANK

وإذا كان الفاصل بين السجلات عبارة عن رمز آخر غير المسافة الخالية مثل الفاصلة (;) أو الشرطة (ـ) فإن الأمر يستخدم هكذا

DELIMITED WITH <*delimiter*>

وتستبدل كلمة delimiterبالحرف أو العلامة المستخدمة كفاصل بين الحقول في الملف المضاف.

ويمكن استبدال اسم الملف بتعبير حرفي موجود في الذاكرة إذا استخدمت (<*expC*>)

الاختلاف عن dBASE III PLUS : تستطيع نقل ملفات صفحة البيانات الالكترونية (spreadsheet) مثل WKS (لوتس ١-٢-٣) أو SYLK أو DIF باستخدام dBASE III PLUS وعندما يتم نقلها إليها تتحول السطور إلى سجلات والأعمدة إلى حقول.

أما Clipper فلا تستطيع التعامل إلا مع ملفات نصية وهي المشار إليها في الأمر بكلمة SDF (System Data Format) ، وهي ملفات مكتوبة بشفرة ASCII وذات أطوال ثابتة.

هذا من ناحية ومن ناحية أخرى لا يشتمل أمر dBASE III PLUS على الاختيار FIELDS أو الاختيار WHILE ولا تستطيع «دي بيس ثري بلاس» التعامل مع الاختيار (<*expC*>) الموجود في أمر «كلبر».

أمثلة :

(١) تستخدم الأوامر التالية لاضافة سجلات من ملف TEMPSALE.dbf إلى نهاية الملف المفتوح باسم SALE.dbf ويشتمل كلا الملفين على حقل منطقي اسمه UPDATED

```
USE SALE
APPEND FROM TEMPSALE FOR .NOT. UPDATED
```

(٢) الأمر التالي يضيف ملف نصي مسجل بشفرة ASCII اسمه BOOK.txt إلى الملف المفتوح LIB.dbf

```
APPEND FROM BOOK.TXT DELIMITED WITH BLANK
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

APPEND BLANK – COPY – READ ( )

الأمر AVERAGE

يحسب المتوسط الحسابي للحقول الرقمية.

الشكل العام:

AVERAGE [expN list] [<*scope*>] [FOR <*condition*>]

[WHILE <*condition*>] [TO <*memvar list*>]

حيث:

<*expN list*>	:	أسماء الحقول الرقمية المطلوب حساب متوسطها الحسابي.
<*scope*>	:	تحدد السجلات التي سيحسب متوسطها الحسابي.
FOR <*condition*>	:	تنفذ الأمر على السجلات التي تتطابق مع الشرط الموجود في الأمر فقط.
WHILE <*condition*>	:	تنفذ الأمر طالما أن الشرط الموجود في الأمر صحيحا.
TO <*memvar list*>	:	يحدد حقول الذاكرة التي ستوضع فيها النتائج.

الشـــرح:

يستخدم هذا الأمر لحساب المتوسط الحسابي للحقول الرقمية في الملف المفتوح. إذا استخدم الأمر بدون أية اختيارات إضافية فإن جميع الحقول الرقمية الموجودة في الملف يتم حساب المتوسط الحسابي لها. فإذا اشتمل الأمر على اسم أو اسماء بعض الحقول فإن الحقول المذكورة أسماءها هي التي يتم حساب متوسطها الحسابي فقط. وكذلك إذا اشتمل الأمر على أحد الاختيارات <*scope*> أو FOR/WHILE فإن السجلات المتطابقة مع الشرط الموجود في الأمر هي التي يتم حساب متوسطها الحسابي.

إذا أردت تخزين النتائج التي تحصل عليها من هذا الأمر فيجب أن تستخدم الاختيار TO <*memvar list*> ثم تذكر أسماء حقول الذاكرة التي ستشتمل على النتائج.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف .

مثـــال :

الأوامـر التـالية تحسب المتـوسط الحسابي لحقل PRICE فقط للسجلات التي تتطابق مع اسم شركة IBM وتضع النتائج في حقل ذاكرة اسمح M_PRICE .

```
USE STOCK
AVERAGE PRICE FOR COMPANY = "IBM" TO M_PRICE
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

COUNT – SUM – TOTAL

الأمر BEGIN SEQUENCE

يمنع توقف التنفيذ عند حدوث خطأ في البرنامج .

الشكل العام:

```
BEGIN SEQUENCE
        Commands...
        [BREAK]
        Commands...
END
Commands...
```

حيث :

BREAK : تنقل التنفيذ بعد أمر END (مثل EXIT داخل DO WHILE) .

الشرح :

هذا الأمر من الأوامر التركيبية التي تسمح بالتحكم في الأخطاء التي تقع أثناء تنفيذ البرنامج . فلكي تنقل التنفيذ إلى الأوامر التي تلي الجزء END استخدم BREAK حتى لا يتوقف التنفيذ وتحصل على رسالة الخطأ . والاختيار BREAK في هذا الأمر يشبه EXIT داخل أمر DO WHILE...ENDDO فعندما تكتشف Clipper هذا الأمر فإن تنفيذ البرنامج ينتقل إلى التعليمة التي تلي END .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثال :

المثال الآتي يطبع تقريراً معد سلفاً إذا كانت الطابعة جاهزة فقط . فإذا لم تكن جاهزة ينقل التنفيذ إلى الأوامر التي تلي الجزء END من الأمر بدون توقف البرنامج أو إظهار رسالة خطأ .

```
BEGIN SEQUENCE
  DO WHILE .T.
     IF ISPRINTER()
        REPORT FORM STRPRT TO PRINT
     ELSE
        BREAK
     ENDIF
  ENDDO
END
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

DO WHILE – RETURN

الأمر CANCEL/QUIT

ينهي البرنامج ويغلق الملفات المفتوحة وينقل التنفيذ إلى نظام التشغيل.

الشكل العام:

CANCEL

QUIT

الشـــرح:

كلا الأمرين يتسبب في إنهاء البرنامج وغلق الملفات المفتوحة والانتقال إلى نظام التشغيل.

الاختلاف عن dBASE III PLUS:

أمر CANCEL داخل dBASE III PLUS ينقل التنفيذ إلى نقطة توجيه الأوامر.

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

RETURN

الأمر CLEAR

يستخدم لأكثر من غرض تبعا للاختيار الموجود بالأمر.

الشكل العام:

CLEAR [ALL/MEMORY/GETS/TYPEAHEAD]

الشرح:

إذا استخدم أمر CLEAR بدون أية اختيارات معه فإنه يمسح الشاشة ويضع المؤشر عند الركن الشمالي السفلي من الشاشة فإذا استخدم مع أمر ...@ فيمكن أن يمسح جزء محدد من الشاشة.

وإليك شرح باقي الاختيارات الموجودة بالأمر.

CLEAR ALL :

يغلق الملفات المفتوحة ويمسح محتويات الذاكرة ويختار المنطقة رقم ١ .

CLEAR GETS :

لا يسمح بتعديل الحقل أو الحقول التي سبق إظهارها بنية التعديل بالأمر GET...@ . ونتيجة لذلك فإن كل الحقول المذكورة بعد GET ستظهر على الشاشة بدون إمكانية التعديل فيها.

CLEAR MEMORY :

يمسح كل محتويات الذاكرة الخاصة PRIVATE والعامة PUBLIC .

CLEAR TYPEAHEAD :

المقصود بكلمة Typeahead هو المحطة التي تتقبل النصوص أو العبارات من المستخدم قبل تنفيذها بواسطة الحاسب وتسمى أحيانا Buffer . ويستخدم هذا الأمر للتأكد أن هذه المحطة خالية من أية نصوص سابقة وذلك قبل الأوامر التي تستقبل معلومات من المستخدم مثل READ أو WAIT أو ACCEPT .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثال :

المثال الآتي يمسح الشاشة أولاً ثم يظهر بيانات الطالب الأساسية بنفس الطريقة التي يتم بها تعديلها أو إدخالها أول مرة بدون صلاحيات التعديل.

```
USE STUDENTS
CLEAR
@  5,5 SAY "First name  :" GET FIRSTNAME
@  7,5 SAY "Family name :" GET LASTNAME
@  9,5 SAY "Address     :" GET ADDRESS
@ 11,5 SAY "Organization:" GET ORGANIZ
@ 13,5 SAY "Phone nunber:" GET PHONE
@ 15,5 SAY "Cource cost :" GET COST
CLEAR GETS
@ 17,20 SAY "Confirm deletion? (Y/N) " TO YN
READ
IF UPPER(YN) = "Y"
   DELETE
   PACK
ENDIF
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

@ – CLOSE – RELEASE

الأمر CLOSE

يغلق ملف/ملفات مفتوحة .

الشكل العام:

CLOSE ALL/ALTERNATE/DATABASES/FORMAT/INDEX/

حيث :

ALL : يغلق جميع أنواع الملفات بدون تأثير على محتويات الذاكرة.

ALTERNATE : يغلق الملف المفتوح من نوع ALTERNATE

DATABASES : يغلق الملف المفتوح من نوع DATABASES والملفات المرتبطة به .

FORMAT : يغلق ملف FORMAT المفتوح .

INDEX : يغلق جميع ملفات INDEX المفتوحة .

الشـــرح:

يغلق هذا الأمر الملف أو الملفات المحددة بعد كلمة CLOSE .

الاختلاف عن *dBASE III PLUS* : لا يستخدم Clipper أمر CLOSE PROCE- DURE

أمثلـــة :

(١) لاغـلاق جميع ملفات قاعدة البيانات المفتوحة وملفات INDEX و FORMAT المتصلة بها استخدم أمر:

CLOSE DATABASES

(٢) لاغلاق جميع الملفات المفتوحة بدون تأثير على محتويات الذاكرة استخدم أمر:

CLOSE ALL

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

CLEAR ALL – CLEAR MEMORY – SET INTEX

SET ALTERNATE – SELECT – SET PROCEDURE – USE

الأمر COMMIT

يحذف البيانات الموجودة في المحطات الانتقالية بعد تعديل بيانات الملفات.

الشكل العام:

COMMIT

الشرح:

استخدم هذا الأمر بعد أية تعديلات تجريها على الملفات للتأكد أن البيانات انتقلت من المحطة الانتقالية (Buffer) إلى القرص. ويشترط أن يكون نظام التشغيل الذي تعمل تحته إصدار 3.3 أو أكثر.

الاختلاف عن *dBASE III PLUS* : غير موجود بها.

مثال:

المثال الآتي يستخدم هذا الأمر بعد تعديل بيانات الطالب ليتأكد أن التعديلات تمت على القرص نفسه وليست على البيانات الموجودة في المحطة الانتقالية (Buffer) .

```
@ 12,5 SAY "Student's address: " GET ADDRESS
@ 14,5 SAY "Student's phone  : " GET PHONE
@ 16,5 SAY "Student's city   : " GET PHONE
READ
COMMIT
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

CLOSE DATABASES

الأمر CONTINUE

يسمح باستمرار البحث عن السجل التالي في الملف المقابل للشرط المحدد في أمر LOCATE .

الشكل العام:

CONTINUE

الشـــرح:

يستخدم هذا الأمر عادة بعد أمر LOCATE للاستمرار في عملية البحث داخل الملف ويسمح بالانتقال إلى السجل التالي في الملف المقابل للشرط المحدد في أمر LO-CATE . ويستمر الأمر في البحث في الملف حتى يجد السجل الذي يتطابق مع الشرط المذكور أو حتى يصل إلى نهاية الملف (في حالة عدم الحصول على السجل المطلوب).

إذا وجدت Clipper السجل الذي تبحث عنه بأمر LOCATE تصبح الوظيفة FOUND() صحيحة (.T.) .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

المثال الآتي يبحث عن الموظف الذي يتقاضى راتباً أعلى من ١٠٠٠ ويظهر اسمه على الشاشة ثم يستمر في البحث عن آخر حتى ينتهي الملف.

```
USE EMNPLOYEE
LOCATE FOR SALARY > 10000
IF .NOT. FOUND()
   RETURN
ENDIF
DO WHILE .T.
  CLEAR
```

```
    @ 2,2 SAY TRIM(NAME1)+TRIM(NAME3)+"Has salary > 10000"
    WAIT "Press a key to find another big salary"
    CONTINUE
    IF FOUND()
       LOOP
    ELSE
       RETURN
    ENDIF
  ENDDO
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

LOCATE – FOUND ( ) – EOF ( ) – FIND

الأمر COPY TO

ينسخ الملف المفتوح إلى ملف آخر.

الشكل العام:

COPY TO <*new file*> /(<*expC1*>) [<*scope*>] [FIELDS <*field list*>]
[FOR <*cond*>] [WHILE <*cond*>] [SDF/DELIMITED
[WITH <*delimiter*>/(<*expC2*>)]]

حيث:

<*new file*>	:	اسم الملف الذي سيتم النسخ إليه.
<*expC1*>	:	اسم حقل الذاكرة أو التعبير الذي سيحل محل اسم الملف.
<*scope*>	:	يحدد السجلات التي سيتم نسخها مثل ALL في حالة الحاجة إلى نسخ جميع السجلات أو NEXT 5 في حالة الحاجة إلى نسخ السجلات الخمسة التالية أو REST في حالة الحاجة إلى نسخ باقي سجلات الملف.
<*field list*>	:	أسماء الحقول المختارة ليتم نسخها.
FOR <*cond*>	:	ينسخ فقط السجلات التي تتطابق مع الحالة المحددة في الأمر.
WHILE <*cond*>	:	تسمح بالنسخ طالما أن الحالة المختارة صحيحة.
SDF	:	تعني أن الملف الذي سيتم النسخ إليه ملف نصي من نوع SDF.
DELIMITED	:	تحدد الفاصل الذي سيستخدم بين الحقول أو الأعمدة.

الشـــرح:

يستخدم أمر COPY TO لنسخ كل السجلات أو بعضها وكذلك كل الحقول أو بعضها من ملف مفتوح إلى ملف آخر إما أن يكون ملف قاعدة بيانات أو ملف نصي

من نوع System Data Format .

إذا لم يحدد الاسم الممتد في الملف المنسوخ إليه فإن قاعدة البيانات تعتبره (dbf.)

الاختلاف عن dBASE III PLUS : لا تتعامل Clipper مع ملفات صفحة البيانات الالكترونية (Spreadsheet) مثل DIF أو SYLK أو WKS وكذلك لا تتعامل مع الاختيار DELIMITED WITH BLANK

مثــال :

المثال التالي يستخدم ملف STUDENTS.dbf لانشاء ملف جديد اسمه RIYADH.dbf يحتوي على الحقول التالية :
FIRSTNAME, LASTNAME, CITY ويحتوي على السجلات التي توجد بها مدينة RIYADH في حقل CITY .

```
USE STUDENTS
COPY TO RIYADH FIELDS FIRSTNAME,LASTNAME,CITY FOR CITY="RIYADH"
```

أما إذا أردت أن تنشىء ملفاً نصياً باسم RIYADH.TXT بحيث يمكنك رؤية محتوياته بأي محرر للنصوص استخدم الأمر بالصورة التالية :

```
COPY TO RIYADH FIELDS FIRSTNAME,LASTNAME,CITY FOR CITY="RIYADH" DELIMITD
```

لرؤية محتويات الملف الجديد اذهب إلى DOS واكتب أمر:

TYPE RIYADH.TXT

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

APPEND FROM – COPY FILE – COPY STRUCTURE

الأمر COPY FILE

ينسخ أي نوع من الملفات.

الشكل العام:

COPY FILE <*Source file*>/(<*expC1*>) TO <*Target file*>/(<*expC2*>)

حيث:

<*Sorrce file*>	:	اسم الملف المطلوب نسخه.
(<*expC1*>)	:	تعبير حرفي أو اسم حقل ذاكرة يدل على اسم الملف.
<*Target file*>	:	اسم الملف المطلوب النسخ إليه.
(<*expC2*>)	:	تعبير حرفي أو اسم حقل ذاكرة يدل على اسم الملف.

الشرح:

يستخدم هذا الأمر لعمل نسخة طبق الأصل من ملف إلى ملف آخر. ويجب تحديد اسم الملف مع الاسم الممتد فإذا كان الملف موجوداً على دليل آخر وجب ذكر اسم الدليل ومشغل القرص. ولا يشترط إغلاق الملف المراد نسخه قبل إصدار الأمر كما هو الحال في dBASE III PLUS .

الاختلاف عن *dBASE III PLUS* : لا تتعامل dBASE مع الاختيار (<*expC*>) ولا تستطيع نسخ الملف إذا كان مفتوحاً.

مثال:

لنسخ ملف اسمه PROG1.prg إلى ملف آخر اسمه PROG2.prg استخدم الأمر التالي:

```
COPY FILE PROG1.PRG TO PROG2.PRG
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

COPY TO – APPEND FROM

الأمر COPY STRUCTURE

ينسخ مواصفات ملف قاعدة بيانات إلى آخر.

الشكل العام:

COPY STRUCTURE [EXTENDED] TO <*new file*>/(<*expC*>) [FIELDS <*field list*>]

حيث:

<*new file*> : اسم الملف الذي سيتم نسخ المواصفات (structure) إليه.

<*field list*> : أسماء الحقول التي سيتم نسخها.

(<*expC*>) : تعبير يدل على اسم الملف.

الشرح:

ينشىء أمر COPY STRUCTURE ملفا جديدا يحمل نفس مواصفات الملف المفتوح ما لم يستخدم الاختيار FIELDS فإذا اخترت FIELDS فسيتم إنشاء ملف يحتوي على الحقول المحددة أسماءها بعد كلمة FIELDS فقط.

الاختيار EXTENDED ينشىء ملفا جديدا يشتمل على ٤ حقول هي: اسم الحقل field name ونوعه type وطوله length وعدد الأرقام العشرية decimal. وتكون محتويات الملف الجديد أو السجلات في الملف الجديد عبارة عن أسماء الحقول field name وأنواعها field type وأطوالها field length وعدد الأرقام العشرية.

الاختلاف عن dBASE III PLUS : لا تستخدم dBASE III PLUS الاختيار (<*expC*>) بدلاً من اسم الملف.

مثال:

(١) ينسخ الأمر التالي مواصفات ملف STOCK.dbf إلى ملف آخر جديد اسمه TEMPSTK.dbf

```
USE STOCK
COPY STRUCTURE TO TEMPSTK
```

(٢) ينسخ الأمر التالي بيانات الحقول التي يشتمل عليها ملف STOCK.dbf ويضعها على ملف جديد اسمه MYSTOCK.dbf.

```
USE STOCK
COPY STRUCTURE EXTENDED TO MYSTOCK
```

ولكي ترى مواصفات الملف الجديد استخدم الأمرين التاليين :

```
USE MYSTOCK
LIST
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

CREATE – CREATE FROM – FIELD ( ) – TYPE ( )

الأمر COUNT

يحسب عدد السجلات الموجودة بالملف.

الشكل العام:

COUNT [<*scope*>] FOR <*condition*>] [WHILE <*condition*>]

[TO <*memvar*>]

حيث:

<*scope*>	:	يحدد السجلات التي سيتم عدها في الملف المفتوح مثل REST أو ALL . . . الخ.
FOR <*condition*>	:	تحسب عدد السجلات التي تتطابق مع الشرط الموجود في الأمر.
WHILE <*condition*>	:	تضمن تنفيذ الأمر (عد السجلات) طالما أن الشرط الموجود بالأمر صحيحا.
TO <*memvar*>	:	تحدد اسم حقل الذاكرة الذي ستوضع به نتيجة الأمر.

الشرح:

يستخدم هذا الأمر لمعرفة عدد سجلات ملف قاعدة البيانات المفتوح. فإذا اشتمل الأمر على أحد الاختيارات <*scope*> أو FLR/WHILE فإن السجلات المتطابقة مع الشرط الموجود بالأمر هي التي يتم عدها فقط. استخدم الاختيار TO <*memvar*> إذا أردت أن تضع ناتج الأمر في حقل الذاكرة المحدد اسمه بعد كلمة TO .

تدخل السجلات المعلمة لأغراض الحذف ضمن عدد السجلات في حالة وضع الدالة DELETED في وضع نعم (ON) وذلك بالأمر SET DELETED ON .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال :

المثـال التالي يستخدم ملف STOCK.dbf لحساب عدد سجلات شركة ATT فقط ووضع النتائج في حقل ذاكرة اسمه M_NO .

```
USE STOCK
COUNT FOR COMPANY = "ATT" TO M_NO
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

RECCOUNT( )

الأمر CREATE

ينشىء ملفاً يشتمل على حقول توصيف الملف (structere extended)

الشكل العام:

CREATE <*file*> /(<*expC*>)

حيث:

File (<*expC*>) : اسم الملف الجديد.

الشرح:

هذا الأمر مشابه تماماً لأمر COPY STRUCTURE EXTENDED فهو ينشىء ملفاً بدون بيانات يشتمل على أربعة حقول هي:

Field_name – Field_type – Field_len – Field_Dec

إلا أنه لا يتطلب وجود ملف قاعدة البيانات.

ويتضح من ذلك أن هذا الأمر يختلف عن أمر CREATE الموجود في dBASE III PLUS في أنه لا يستدعي شاشة تصميم الملف. ويمكن الاستفادة من هذا الأمر في إعطاء المستفيد من النظام الفرصة في تحديد مواصفات الملف كما يتضح ذلك من المثال التالي.

***الاختلاف عن dBASE III PLUS*:** يستدعي أمر CREATE dBASE III PLUS شاشة كاملة لادخال مواصفات الملف.

مثـال:

المثال التالي يتيح للمستفيد من النظام تعديل مواصفات ملف SAMPLE.dbf .

```
CREATE SAMPLE
USE SAMPLE
APPEND BLANK
?RECNO()                        اجابة كليبر : 1 &&
@ 10,10 say "Name" GET FIELD_NAME
@ 12,10 say "Type" GET FIELD_TYPE
@ 14,10 say "Len." GET FIELD_LEN
@ 16,10 say "Dec." GET FIELD_DEC
READ
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

CREATE FROM – COPY STRUCTUR EXTENDED

الأمر CREATE FROM

ينشىء ملفاً جديداً مستخدماً محتويات ملف من نوع STRUCTURE EXTENDED .

الشكل العام:

CREATE <*new file*> /(<*exp1*>)

FROM <*structure extended file*>/(<*expC2*>)

حيـــث:

<*new file*>/ (<*expC1*>) : اسم الملف الجديد.

Structure extended : اسم ملف سبق إنشاؤه بأمر

<*file*> : COPY STRUCTURE EXTENDED

<*expC2*> : اسم ملف سبق إنشاؤه.

الشـــرح:

يستخدم هذا الأمر الحقول الأربعة التي يتم وصفها في الملف الذي ينشأ بأمر COPY STRUCTURE EXTENDED

لانشاء ملف جديد. ويمكنه التعامل مع هذه الحقول بأي ترتيب والملف الذي ينشأ بأمر COPY STRUCTURE EXTENDED

يتكون من ٤ حقول بالشكل الآتي:

Field	Name	Type	Length	Dec.
1	Field_name	C	10	
2	Field_type	C	1	
3	Field_lan	N	3	0
4	Field_dec	N	3	0

ويمكن الاستفادة من هذا الأمر في تحسين مواصفات هذا الملف. فمثلاً طول الحقل الحرفي كما هو واضح لا يقبل أكثر من ٣ خانات أي ٩٩٩ ولما كان «كلبر» يستطيع التعامل مع حقول حرفية تصل إلى ٣٢ك.ب. فلابد أن تضع في مواصفات الملف طول الحقل ولابد من تعديل الرقم 3 الموجود تحت Length في الجدول السابق ليقبل ٥ خانات.

الاختلاف عن *dBASE III PLUS* : باستخدام Clipper تستطيع زيادة طول الحقل الحرفي عن ٩٩٩ كما يمكنك تعديل محتويات ملف STRUCTURE EXTENDED .

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

CREATE – COPY STRUCTURE EXTENDED

الأمر DECLARE

ينشىء مصفوفة أو أكثر ذات بُعد واحد.

الشكل العام:

DECLARE *<arrayname1>* [*<expN1>*] [,*<arrayname2>* [*<expN2>*]]...

حيــث:

<arrayname1>	:	اسم المصفوفة المزمع إنشاؤها.
(*<expN1>*)	:	عدد عناصر المصفوفة الأولى.
<arrayname2>	:	اسم المصفوفة الثانية.
<expN2>	:	عدد عناصر المصفوفة الثانية.

هـام: الأقواس التي تحيط بالمعامل <expN> في هذا الأمر هكذا [ ] لا تعني أن ما بداخلها اختيارياً كما هو الحال في باقي أوامر قاعدة البيانـات. ولكنهـا جزء من تكوين الأمر نفسه، ولذلك لابد من كتابتها. وهذا استثناء وحيد من قاعدة استخدام هذه الأقواس.

الشـــرح:

يسمح أمر DECLARE بإنشاء مصفوفة ذات بُعد واحد أو أكثر من مصفوفة. وتتكـون المصفـوفة الواحدة من عدد من العناصر لا يزيد عن ٢٠٤٨ عنصراً وتخزن بالذاكرة وتعد بحقل ذاكرة واحد ويعامل هذا الحقل معاملة حقول الذاكرة الخاصة (PRIVATE) وتختلف المصفوفة وعناصرها الموجودة بالذاكرة عن حقول الذاكرة -Mem) (ory variables الأخرى في أنها لا تحفظ على ملف خارجي (.MEM).

وبعـد إنشـاء المصفـوفة (بالاسم وعدد العناصر) يجب تخصيص قيم لعناصر المصفوفة باستخدام أمر STORE أو = اللذان يستخدمان لتخصيص حقول الذاكرة.

لكي تعرف عدد عناصر المصفوفة استخدم الوظيفة ( )LEN مستخدماً اسم المصفوفة كمعامل للوظيفة.

ويمكن أن تكون عناصر المصفوفة الواحدة من أنواع مختلفة (حرفية أو رقمية أو تاريخية أو منطقية). وإذا أردت أن تعرف نوع عنصر داخل المصفوفة. خصص هذا العنصر لحقل ذاكرة ثم اسأل عنه بالوظيفة ( )TYPE .

ويمكن استقبال المصفوفة وعناصرها كمعطيات (parameters) داخل البرامج والاجراءات والوظائف الخاصة.

الاختلاف عن *dBASE III PLUS* : لا تتعامل dBASE III PLUS مع المصفوفات.

أمثلة :

١) لانشاء مصفوفة ذات بُعد واحد اسمها MONTHS تتكون من ١٢ عنصراً استخدم الأمر التالي :

```
DECLARE MONTHS [12]
```

٢) ولانشاء مصفوفتين كلتاهما ذات بُعد واحد استخدم الأمر بالشكل الآتي :

```
DECLARE MONTHS [12], SALES [3]
```

وإذا أردت تخصيص القيمة "Feb." للعنصر الثاني من المصفوفة MONTHS استخدم الأمر التالي:

```
STORE "Feb." TO MONTHS[2]
```

٣) المثال الآتي ينشىء مصفوفة ويخصص قيماً لعناصرها ثم يظهر هذه العناصر

```
DECLARE MONTHS[12]
MONTHS[1] = "Jan."
MONTHS[2] = "Feb."
MONTHS[3] = "Mar."
MONTHS[4] = "Apr."
? MONTHS[3]                        && Mar. اجابة كليبر:
```

٤) المثال التالي يشير إلى المصفوفات باستخدام Macro

```
NAME = "SALE"
ELEMENTS = 3
NAME_LEN = "SALE[1]"
*
DECLARE &NAME[ELEMENTS]        && SALE انشىء مصفوفة ذات ثلاثة عناصرباسم
&NAME_LEN = 15                 && ضع الرقم 15 فى أول عنصر
&name[3] = "Year"              && ضع كلمة Year فى العنصر الثالث:

DECLARE &NAME_LEN
```

أما الاستخدام غير المسموح به في هذا المثال فهو كما يلي:

```
DECLARE &NAME_LEN
```

لأن الأقواس [ ] لا يمكن استخدامها داخل Macro

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

PUBLIC – PRIVATE – STOR – ADEL( ) – ADIR( ) – AFILL( ) –
AINS( ) – ASCAN( ) – ASORT( )

الأمر DELETE

يعلم السجلات لغرض حذفها من الملف.

الشكل العام:

DELETE [<*scope*>] [FOR <*condition*>] [WHILE <*condition*>]

حيث:

<scope>	:	**تحدد السجلات التي سيتم حذفها (مثل all أو rest .... الخ).**
FOR <condition>	:	**تحذف السجلات التي تتطابق مع الشرط الموجود في الأمر فقط.**
WHILE <condition>	:	**تسمح بحذف السجلات طالما أن الشرط المذكور في الأمر صحيحا.**

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

الشـــرح:

يضع هذا الأمر علامة أمام السجلات التي ترغب في حذفها من الملف. وعادة توضع هذه العلامة أمام السجل الذي يقف عنده المؤشر داخل الملف إذا استخدم هذا الأمر بدون اختيارات أخرى.

فإذا اشتمل الأمر على اختيار <Scope> أو FOR/WHILE <Condition> فإن السجلات التي تتطابق مع الشرط تعلم جميعها بغرض حذفها فيما بعد.

وتعرف السجلات المعلمة لأغراض الحذف بظهور علامة "*" على شمال كل سجل مع أوامر LIST وDISPLAY . السجلات المعلمة لغرض الحذف تضع .T. في الوظيفة DELETED .

مثـــال:

لوضع علامة أمام السجل الأخير في الملف المفتوح بنية حذفه فيما بعد استخدم الأوامر التالية:

```
GO BOTTOM
DELETE
? DELETED()              && .T. :  الإجابة
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

DELETED ( ) – PACK – RECALL – SET DELETED – ZAP – EMPTY

الأمـر DIR

يظهر اسماء الملفات الموجودة على القرص أو الدليل المخصص معك .

الشكل العام للأمر:

DIR *[drive* > :] [<*path*>][<*skeleton*>]

حيـث :

drive : اسم مشغل القرص إذا كنت ستظهر ملفات غير موجودة على القرص المخصص معك .

path : الطريق الذي يجب تسلكه قاعدة البيانات في البحث عن الملفات إذا لم تجد الملف في الدليل المخصص .

skeleton : مجموعة من الحروف (علامة الاستفهام ? تدل على غياب حرف واحد . أما العلامة * فتدل على غياب مجموعة حروف) .

الشـــرح:

يظهر الأمر أسماء الملفات وعدد السجلات الموجودة في كل ملف وتاريخ آخر تعديل في الملف وحجم كل ملف بالحروف (Bytes) . وفي النهاية إجمالي عدد الملفات المعروضة وإجمالي حجمها وإجمالي المساحة المتبقية على القرص ما لم تستخدم أحد الاختيارين Path, drive أو كليهما .

الاختيار Skeleton يتيح لك إظهار ملفات من أنواع أخرى غير dbf . ويظهر لك أيضا معها إجمالي عدد الملفات المعروضة وإجمالي المساحة المتبقية على القرص .

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف .

أمثلـــة:

١ ـ لاظهـار أسـماء ملفات قاعدة البيانات الموجودة على القرص والدليل المخصص معك استخدام الأمر

DIR

٢ ـ ولاظهار أسماء جميع ملفات البرامج (.prg) على الدليل المسمى DBMS الموجود على الوحدة C ادخل الأمر

DIR C: \ DBMS \ *.prg

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

ADIR ( )

الأمر DISPLAY

يظهر محتويات الملف.

الشكل العام للأمر:

DISPLAY [<*scope*>] [<*expression list*> [FOR <*condition*>]
[WHILE <*conditon*>] [OFF] [TO PRINT] [TO FILE <*file*>/(<*expC*>)]

حيـــث:

<*scope*>	:	تحدد السجلات التي سيتم إظهارها من الملف (مثل ALL أو REST . . . الخ).
<*expression list*>	:	أسماء الحقول المطلوب إظهارها مفصولة بعلامة «,».
FOR <*condition*>	:	تظهر السجلات المتطابقة مع الشرط الموجود في الأمر فقط.
WHILE <*condition*>	:	تسمح بتنفيذ الأمر طالما أن الشرط الموجود في الأمر صحيحا.
OFF	:	تلغي رقم السجـل (Record #) الـذي يظهـر على شمال السجل.
TO PRINT	:	توجه المخرجات إلى الطابعة.
TO FILE	:	توجيه المخرجات إلى ملف نصي مكتوب بشفرة ASCII.

الشـــرح:

يستخدم هذا الأمر لاظهار بيانات السجل الذي يقف عنده المؤشر داخل ملف قاعدة البيانات. فإذا اشتمل الأمر على أحد الاختيارات <*scope*> أو FOR/WHILE فإن جميع السجلات المتطابقة مع الشرط الموجود في الأمر يتم إظهارها.

وأيضا إذا اشتمل الأمر على الأختيار *<expression list>* فإن الحقل أو الحقول المذكورة فقط هي التي تظهر.

الاختيار OFF يلغي ظهور الرقم التلقائي الذي تخصصه قاعدة البيانات للسجلات والذي يظهر عادة على شمال السجل تحت عنوان # Record .

الاختيار TO PRINT يوجه ناتج الأمر إلى الطابعة بالاضافة إلى الشاشة. والاختيار TO FILE ينشىء ملفا نصيا (.TXT.) .

إذا كان المؤشر يقف عند نهاية الملف وأصدرت أمر DISPLAY فلن يظهر شيء سوى أسماء الحقول. ولذلك يجب أن تعيد تحريك المؤشر لتظهر لك بيانات السجلات.

السجلات المعلمة لأغراض الحذف تظهر أمامها علامة * على يمين السجل.

الاختلاف عن dBASE III PLUS : لا يوجد في dBASE III PLUS الاختيار TO FILE . واستخدام هذا الأمر في Clipper لا يعطي فرصة لتوقف التنفيذ عند امتلاء الشاشة وانتظار ضغط أحد المفاتيح للاستمرار في العرض.

مثـــال:

المثال التالي يستخدم ملف STOCK.dbf لاظهار الحقول:

ACCOUNTNO – NO_SHARES – PRICE

بالاضافة إلى NO_SHARES * PRINCE بالنسبة للسجلات التي تحتوي على S في حقل TYPE . مع توجيه المخرجات إلى الطابعة أيضا وإلغاء الرقم الذي يظهر على شمال السجل:

```
DISPLAY OFF ACCOUNTNO,NO_SHARES,PRICE,NO_SHARES*PRICE FOR TYPE="S" TO PRIT
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

LIST – SET DELETED

الأمر DO

يستدعي برنامجا للتنفيذ.

الشكل العام:

DO <*procedure name*> WITH <*parameter list*>

حيث:

<*procedure name*> : اسم البرنامج أو الاجراء (Procedure) المطلوب استدعاؤه.

<*parameter list*> : المعطيات التي سيقبلها البرنامج.

الشرح:

يستخدم هذا الأمر لاستدعاء برنامج أو إجراء.

وتبدأ قاعدة البيانات بالبحث عن اسم البرنامج المذكور (.prg) في الدليل أو القرص المخصص. وتقوم بترجمته ولذلك فيجب أن يشتمل الأمر على اسم الدليل أو القرص الذي ستبحث داخله قاعدة البيانات عن اسم البرنامج في حالة وجود البرنامج على دليل أو قرص آخر غير المخصص معك. أما إذا كان المطلوب استدعاء إجراء فإنها تبحث عنه أمام كلمة PROCEDURE .

وعندما ينتهي تنفيذ البرنامج (أو الاجراء) ترجع قاعدة البيانات إلى الأمر التالي لأمر DO في البرنامج السابق.

إذا اشتمل الأمر على الاختيار WITH فمعنى ذلك أنك ستستخدم معطيات (Parameters) ليتم تنفيذها داخل هذا البرنامج وفي هذه الحالة يجب أن يحتوي البرنامج المطلوب للتنفيذ على الأمر PARAMETERS كأول أمر داخل البرنامج .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــال :

المثال التالي يظهر نتيجة تنفيذ إجراء باستخدام معطيات

```
PROCEDURE NET
  PARAMETERS SALARY,TAX
  NETSALE = SALARY - SALARY*TAX
RETURN
*
DO NET WITH 7000,07
?NETSALE                        && 6510  اجابة كلبر:
```

CLIPPER.LIB

الأوامر ذات الصلة :

PARAMETERS - PRIVATE - PROCEDURE - PUBLIC - SET PROCEDURE

الأمــر DO CASE

يعني تنفيذ حالة من عدة حالات متاحة للبرنامج .

الشكل العام:

```
DO CASE
    CASE <condition>
            <commands>
    CASE <condition>
            <commands>
    [OTHERWISE
            <commands>]
ENDCASE
```

حيــــث:

<condition> : تعبيرا منطقيا يصف حالة ما تحتمل الصواب والخطأ.

<commands> : مجموعة الأوامر التي ستنفذ في حالة وقوع الحالة صحيحة.

الشـــــرح:

يستخـدم هذا الأمر داخل البرنامج لتنفيذ مجموعة الأوامر التي تلي أول حالة يكتشف البرنامج أنها صحيحة من بين مجموعة الحالات المذكورة في الأمر. ويستمر تنفيذ الأوامـر التي تلي الحـالـة الصحيحة (CASE <*Condition*>) حتى تجد قاعدة البيانات إحدى الكلمات التالية:

CASE <*Condition*> أو OTHERWISE أو ENDCASE

فينتقل التنفيذ إلى الأمر التالي لعبارة ENDCASE .

ومعنى ذلك أن حالة واحدة فقط من الحالات المذكورة في الأمر هي التي تنفذ وبالتالي فإن جميع الحالات الأخرى تهمل حتى لو كانت إحداها أيضا صحيحة.

فإذا كانت جميع الحالات المذكورة في الأمر خاطئة واشتمل الأمر على عبارة OTHERWISE ومعناها وإلا فإن الأوامر التي تلي هذه العبارة هي التي تنفذ.

أمـا إذا لم يشتمـل الأمر على عبارة OTHERWISE ولم تقع أي من الحالات المـذكـورة صحيحـة فلن تنفـذ أية تعليمات من تلك المـوجـودة داخـل أمـر DOCASE..ENDCASE وسينـتـقـل تنـفيذ الـبرنـامـج إلى أول أمـر يلي عبـارة ENDCASE.

جميـع الأوامـر الـتي تسـتـخـدم عبـارتـين مثـل IF..ENDIF أو DO CASE...ENDCASE أو DO WHILE...ENDDO يمكن أن تستخدم بالتداخل داخل أمر DO CASE.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

يستخدم المثال التالي أمر DO CASE...ENDCASE لحساب مقدار الضرائب المستحقة.

```
DO CASE
   CASE SALARY >= 2000 .AND. SALARY < 3000
      TAX = SALARY * .7
   CASE SALARY >= 3000 .AND. SALARY < 5000
      TAX = SALARY * .10
   CASE SALARY >= 5000
      TAX = SALARY * .15
   OTHERWISE
      TAX = 0
ENDCASE
```

وفي هذا المثال إذا وقعت الحالة الأولى صحيحة أي إذا كان الراتب (SALARY) أكبر من أو يساوي ٢٠٠٠ وفي نفس الوقت أقل من ٣٠٠٠ فإن الأمر الذي سينفذ هو:

TAX = SALARY *.7

وبالتالي ستهمل باقي الحالات ولن تنفذ. أما إذا وقعت هذه الحالة غير صحيحة أي إذا كان الراتب لا يقع في الحدود من ٢٠٠٠ إلى ٣٠٠٠٠ ريال فإن هذا الأمر لن ينفذ.

وبنفس الطريقة إذا وقعت الحالة الثانية صحيحة فإن الأمر الذي سينفذ هو:

TAX = SALARY *.10

أما إذا كان الراتب يقع ضمن شريحة غير الشرائح التي ذكرت بالأمر (أي إذا كان الراتب أقل من ٢٠٠٠) فإن الأمر الذي يلي عبارة OTHERWISE هو الذي ينفذ. لأن كل الحالات المذكورة تكون في هذه الحالة خاطئة.

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

IF – DO – DO WHILE – IF ( ) / IIF ( )

الأمر DO WHILE

ينشىء دوارة داخل البرنامج ينتج عنها تكرار مجموعة من الأوامر.

الشكل العام:

```
DO WHILE <condition>
        <commands>
        [EXIT]
        [LOOP]
ENDDO
```

حيث:

<condition>	:	تعبيرا منطقيا يصف حالة ما تحتمل الصواب والخطأ.
<commands>	:	مجموعة الأوامر التي سيتكرر تنفيذها طالما أن الحالة المذكورة في الأمر صحيحة.

الشرح:

هذا الأمر يستخدم فقط داخل البرنامج وهو ينشىء دوارة داخل البرنامج ينتج عنها تكرار مجموعة الأوامر الواقعة بين DO WHILE وENDDO (*<Commands>*) طالما أن الحالة المذكورة في الأمر (*<Condition>*) صحيحة. أما إذا كانت الحالة المذكورة في الأمر خاطئة فإن التنفيذ ينتقل إلى الأمر التالي لعبارة ENDDO وتهمل جميع الأوامر الواقعة بين DO WHILE وENDDO.

ويستخدم هذا الأمر غالباً لانشاء دورة تنفذ عددا من المرات غير محدود فإذا كان عدد مرات التنفيذ محدود استخدم أمر FOR...NEXT .

إذا اشتمل الأمر DO WHILE...ENDDO على الأمر LOOP داخل الدوارة فإن التنفيذ ينتقل مباشرة إلى أول الدوارة أي إلى أمر DO WHILE ويتم تقييم الحالة المشروحة في الأمر مرة ثانية وبالتالي يتكرر تنفيذ الأوامر التي تلي DO WHILE . وبمجرد انتقال التنفيذ إلى أمر DO WHILE تنفيذا لأمر LOOP فإن كل الأوامر التي تلي أمر LOOP والموجودة داخل الدوارة تهمل.

وإذا اشتملت الدوارة (DO WHILE...ENDDO) على أمر EXIT فإن تنفيذ البرنامج ينتقل مباشرة إلى خارج الدوارة أي إلى الأمر الذي يلي عبارة ENDDO .

يجوز أن تشتمل الدوارة على دوارة أخرى أو دوارات داخلية كما يجوز أن تشتمل على الأوامر التركيبية التي تستخدم عبارتين مثل:

DO CASE... ENDCASE – IF...ENDIF

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

يستخدم المثال التالي الدوارة (DO WHILE) لاظهار الشفرة الأمريكية لتبادل المعلومات (ASCII) والحرف المقابل لكل شفرة ويجب أن يكتب داخل برنامج أو يكتب كبرنامج مستقل.

```
I = 0
DO WHILE I < 256
   ? "The ASCII value" + LTRIM(STR(I)) + "Equal character" + CHR(I)
   I = I+1
ENDDO
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

LOOP – EXIT – RETURN – IF – FOR...NEXT

الأمـر EJECT

يسمح بقفز صفحة من صفحات الورق المركب على الطابعة.

الشكل العام:

EJECT

الشـــرح:

يسمح هذا الأمر بقفز صفحة من صفحات الورق المركب على الطابعة ويسمح بالطباعة ابتداء من أول الصفحة التالية. وبالتالي فإن كلا من الوظيفتين PROW() و PCOL() تصير صفرا.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

المثال التالي يطبع العبارتين في صفحتين متتاليتين.

```
SET PRINT ON
? "First page"
EJECT
? "Second page"
SET PRINT OFF
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

SETPRC()

الأمــر ERASE /DELETE FILE

يحذف الملف المحدد في الأمر من الدليل.

الشكل العام:

ERASE <*file.extension*> / DELETE FILE <*file.extension*>

حيــث:

<file.extension> : اسم الملف المطلوب حذفه.

الشــرح:

يحذف هذا الأمـر الملف المحـدد في الاختيار <file.extension> . ويجب أن يشتمل اسم الملف على الاسم الممتد (extension) ويجب إغلاق الملف المراد حذفه قبل إصدار أمر الحذف.

إذا كان الملف المطلوب حذفه موجودا على دليل أو قرص آخر فيجب أن يتضمن الأمر اسم الدليل أو اسم القرص أو كليهما معا.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثــال:

لحذف ملف اسمه TEMPSAL.dbf من الوحدة والدليل المخصصين معنا يجب إدخال الأمر التالي:

ERASE TEMPSAL.dbf

ولحذف ملف اسمه ISALE.ndx من وحدة القرص A :

ERASE A:\ISALE.ndx

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

CLOSE – USE

الأمـــر FIND

يضع المؤشر عند أول سجل يتطابق مع العبارة المذكورة في الأمر داخل ملف مفهرس.

الشكل العام:

FIND <*character string*>/<*expC*>

حيــــث:

<*character string*> : أي بيانات حرفية.

<*expC*> : عبارة حرفية.

الشــــرح:

يستخدم هذا الأمر للبحث في ملف قاعدة البيانات المفتوح بشرط أن يكون سبق فهرسته. ويضع المؤشر على أول سجل موجود بالملف يشتمل على العبارة الموجودة بالأمر. وعادة يتم البحث داخل الحقل المختار للفهرسة فقط (Index Key) .

وعادة يتم البحث داخل الملف المفهرس عن العبارة المتطابقة مع العبارة الموجودة في الأمـــر. فإذا وجـــدتهـــا قاعـــدة البيانـــات اعتـــبرت أن العبـــارة موجـــودة وأوقفت المـؤشر عنـــد هذا الســـجـــل فمثـــلا إذا كنت تبحث عن كلمـــة ATT في حقـــل COMPANY وأدخلت الأمر بهذه الصورة:

FIND "A"

فستضــع المؤشر عند أول سجل يشتمل على الكلمة التي تبدأ بحرف A فإذا وجدت AST مثلا اعتبرت أن الشرط قد تحقق وأوقفت البحث في حين أننا نبحث عن ATT .

فإذا كنت تريد التطابق التام أثناء البحث أي إذا كنت تريد أن يستمر البحث حتى تجد كلمة ATT كلها فيجب أن تضع أمر EXACT في وضع ON هكذا

SET EXACT ON

وإذا كان البحث عن محتويات ذاكرة حرفية فيجب أن تضع الوظيفة & قبل اسم حقل الذاكرة.

إذا وقع الشرط صحيحا وتم الحصول على السجل المطلوب فإن المؤشر ينتقل إلى هذا السجل وتصبح الوظيفة FOUND( ) صحيحة (.T.) . أما إذا لم يوجد السجل المطلوب داخل الملف فإن المؤشر يوضع في نهاية الملف وتصبح الوظيفة EOF( ) صحيحة (.T.) . أما الوظيفة FOUND( ) فتصبح خاطئة (.F.) .

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال :

المثال الآتي يستخدم أمر FIND بطريقتين. وهو يبحث عن السجل الذي يشتمل على كلمة IBM في حقل COMPANY والنتيجة في الحالتين واحدة.

```
USE STOCK INDEX ICOMPANY
FIND "IBM"
?RECNO()                    && 1  اجابة كلبر:
*
*
STORE "IBM" TO M_COMP
FIND &M_COMP
?RECNO()                    && 1  اجابة كلبر:
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

INDEX – SEEK – SET INDEX – SET ORDER – EOF( ) – FOUND( ) – SET EXACT

الأمر FOR...NEXT

ينشىء دوارة تنفذ عدداً محدداً من المرات.

الشكل العام:

```
FOR <memvar> = <expN1> TO <expN2>
[STEP <expN3>]
        <Commands>
        [EXIT]
        <Commands>
NEXT
```

حيث:

<memvar>	:	حقل الذاكرة الذي سيحدد تنفيذ الدوارة.
<expN1>	:	القيمة الأولية لحقل الذاكرة.
<expN2>	:	الحد الأعلى.
[STEP <exp3>]	:	القيمة التي سيزيد بها حقل الذاكرة في كل مرة تنفذ فيها الدوارة.
EXIT	:	تنقل التنفيذ خارج الدوارة بعد أمر NEXT .

الشرح:

هذا الأمر مشابه لأمر DO WHILE...ENDDO وهو مشابه لنظيره في لغات «سي» أو «بيسك» أو «باسكال»، ويستخدم بصفة خاصة عندما تحتاج لتنفيذ دوارة لعدد محدد من المرات. لأنها تستخدم أوامر أقل وأسهل في فهمها وأسرع في تنفيذها. ويتم تقييم التعبيرات الموجودة في الأمر في كل مرة يتم فيها إعادة تنفيذ الدوارة. إذا لم تحدد الاختيار STE P فستخصص له «كلبر» القيمة ١ .

وأشهر استخدام لهذا الأمر لتخصيص قيم للمصفوفات كما سيتضح من المثال التالي.

الاختلاف عن dBASE III PLUS : لا يوجد بها.

مثـــال :

المثال التالي يضع الأرقام من ١ إلى ١٠ داخل مصفوفة تتكون من ١٠ عناصر.

```
DECLARE YEARS[10]          && YEARS انشيء مصفوفة ذات عشرة عناصر باسم
FOR AA=1  TO 10            && كرر الاوامر التالية بعدد 10 مرات
   CLAER
   YEAR=0
   @ 12,2 SAY "Enter new year:" GET YEAR
   READ
   YEARS[AA] = YEAR
NEXT
```

وهو مساوٍ للمثال التالي :

```
DECLARE YEARS[10]
YEAR=0
DO WHILE YEAR<=10
   CLEAR
   @ 12,2 SAY "Enter new year" GET YEAR
   READ
   YEARS[AA] = YEAR
   YEAR=YEAR+1
ENDDO
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة:

DO CASE – DO WHILE – IF

الأمر FUNCTION

ينشىء وظيفة خاصة تستخدم مثل وظائف «كلبر».

الشكل العام:

FUNCTION <*function name*>

<*Commands*>

RETURN <*exp*>

حيث:

<*function name*>	:	الاسم المختار للوظيفة والذي سيتم استدعائها به.
<*exp*>	:	القيمة التي ستخصص للوظيفة في نهاية تنفيذها.
<*Commands*>	:	مجموعة الأوامر التي تعطينا في النهاية <*exp*> .

الشرح:

يعتبر هذا الأمر من أهم الأوامر التي أضافتها Clipper والتي لم تكن موجودة في dBASE III PLUS وهو ينشىء وظيفة خاصة شأنها شأن وظائف «كلبر» المعروفة إلا أنها ينشئها المبرمج لتقوم بعمل محدد ولذلك يقال عنها User Defined Function أو وظائف خاصة. وبمجرد إنشاء وظيفة خاصة بك يمكنك استخدامها في أي مكان داخل البرنامج لتنفيذ بعض الأوامر لأنها تقوم مقام برنامج فرعي أو مجموعة من الأوامر (Sub modules) يتم استدعائها للتنفيذ داخل نفس البرنامج أو من برنامج آخر أو إجراء آخر.

وقد توضع الوظيفة الخاصة في بداية البرنامج كما قد توضع في نهاية البرنامج أو قد توضع بين ملفات الاجراءات (procedure file) أو وظائف خاصة أخرى. وعند استدعاء إحدى الوظائف الخاصة للتنفيذ فإنها تستقبل بيانات من خارج مجموعة الأوامر

التي تشتمل عليها لتحل محل المعطيات (parameters) الموجودة بها. وتقوم «كلبر» بتنفيذ الأوامر التي تشتمل عليها الوظيفة باستخدام البيانات الداخلة إليها وفي النهاية تخصص قيمة للوظيفة الخاصة وهي ما يطلق عليه <*exp*> (راجع الفصل الخامس لمزيد من التفصيلات).

الاختلاف عن *dBASE III PLUS*: غير موجود بها.

مثـــال:

المثال التالي ينشىء وظيفة خاصة باسم CENTER لضبط عبارة وسط السطر بعد حذف البيانات الموجودة على نفس السطر.

```
FUNCTION CENTER                   && CENTER اسم الوظيفة:
  PARAMETER STRING,LENGTH         && ستقبل الوظيفة قيمتين من البرنامج الذى
                                  && سيتدعيها
  PRIVATE LSPACE,RSPACE,CLC_VAL && يجب ألا تتغير محتويات هذه الحقول حتى
                                  && لو وردت أسماؤها فى برامج أخرى
  * تقوم الأوامر التالية بحساب الفراغات التى ستظهر يمين أو يسار العبارة
  LSPACE  = INT((LENGTH - LEN(STRING))/2)
  RSPACE  = LENGTH - LSPACE - LEN(STRING)
  CLC_VAL = SPACE(LSPACE) + STRING + SPACE(RSPACE)
RETURN  CLC_VAL
```

ولكي تستخدم هذه الوظيفة يجب استدعائها مثل وظائف «كلبر» الأخرى. ويجب إعطاء الوظيفة قيمة للتعويض عن String

```
*
@  2,5 PROMPT CENTER("Adding",20)
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

PROCEDURE

الأمـر GO/GOTO

يضع المؤشر على سجل معين داخل الملف.

الشكل العام:

GO/GOTO <expN>

GO/GOTO BOTTOM/TOP

حيــث:

expN : رقم السجل المطلوب تحريك المؤشر إليه.

الشــرح:

يضع الأمر المؤشر عند سجل معين داخل ملف قاعدة البيانات المفتوح. وإليك شرح الصيغ المستخدمة مع الأمر:

GO <expN> : تضع المؤشر عند السجل المحدد رقمه في كلمة <ExpN>.

GO TOP/BOTTOM : تضـع المؤشـر إمـا عند بداية الملف إذا استخدم بصيغة GO TOP أو عنـد نهاية الملف إذا استخـدم بصيغـة GO BOTTOM ويجـب الانتبـاه إلى أن الملف إذا كان مفهرسا فإن بداية أو نهاية الملف تعني أول أو آخر سجل في الملف المفهرس وليس الملف الأصلي لقاعدة البيانات.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

استخدم ملف STOCK.dbf في المثال التالي:

المثال يحرك المؤشر مرة إلى سجل معين وأخرى إلى أول الملف أو آخر الملف.

```
USE STOCK
GO 4
? RECNO()              && 4  الإجابة:
GO TOP
? REECNO()             && 1  الإجابة:
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SKIP – RECNO( )

الأمر IF

يسمح بتنفيذ أمر أو مجموعة أوامر بشرط معين.

الشكل العام:

```
IF <condition>
      <commands1>
[ELSE IF <Condition>
      <commands2>]
[ELSE
      <commands1>]
ENDIF
```

حيث:

<condition>	:	تعبيرا منطقيا يحتمل الصواب والخطأ.
<commands>	:	الأوامر المطلوب تنفيذها إذا وقع الشرط صحيحا.
ELSEIF	:	تحدد الشرط الذي سيتم اختباره إذا وقع الشرط الأول خاطئا.
ELSE	:	معناها إذا لم يتحقق الشرطان السابقان.

الشـــرح:

يستخدم هذا الأمر داخل البرنامج لتنفيذ مجموعة أوامر (<commands1>) إذا تحقق الشرط الموجود بالأمر (<condition>). أما إذا لم يتحقق الشرط الموجود في الأمر واشتمل الأمر على الاختيار ELSEIF فإن الأوامر التي تلي عبارة ELSEIF هي التي تنفذ (<commands2>).

فإذا لم يتحقق أيا من الشرطين الموجودين أمام IF أو ELSEIF فإن الأوامر التي تلي عبارة ELSE هي التي تنفذ. أما إذا لم يتحقق أيا من الشرطين الموجودين بالأمر ولم يشتمل الأمر على عبارة ELSE فإن جميع الأوامر الموجودة بين IF وENDIF ستهمل وسينتقل التنفيذ مباشرة إلى الأمر الذي يلي عبارة ENDIF.

ويجوز أن يشتمل الأمر على أكثر من IF..ENDIF كما يجوز أن يشتمل على الأوامر التركيبية أي التي تستخدم عبارتين مثل:

DO WHILE..ENDDO – DO CASE...ENDCASE – IF...ENDIF

وفي هذه الحالة يجب استخدام الأمر بعناية شديدة.

إذا اشتمل الأمر على أكثر من IF فإن عبارة ELSE دائما تشير إلى أقرب IF إليها أي إلى آخر IF استخدمت.

الاختلاف عن *dBASE III PLUS* : لا تتعامل dBASE III PLUS مع الاختيار ELSEIF .

مثـــال:

المثال التالي يوضح كيفية استخدام الأمر لاتخاذ أكثر من قرار

```
USE EMPLOYE
IF DEGREE = 5
  TAX = SALARY * .05
  ELSEIF DEGREE = 6
    TAX = SALARY * .06
    ELSEIF DEGREE < 5
      TAX = SALARY * .02
ELSE                             && 6 < الدرجة
  TAX = SALARY * .10
ENDIF
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

DO CASE – IIF

الأمـــر INDEX ON

ينشيء ملفا مفهرسا لملف قاعدة البيانات طبقا لبيانات حقل/ حقول معينة.

الشكل العام:

INDEX ON <*key expression*> to< > <*index file*>/(<*expC*>)

حيـــث:

<key expression> : اسم الحقل المطلوب فهرسة الملف طبقا لبياناته.

<index file>/(<*expC*>) : اسم ملف الفهرس الجديد.

الشـــرح:

ينشيء هذا الأمر ملفا جديدا مفهرسا من ملف قاعدة البيانات المفتوح ويأخذ الملف المفهرس اسما ممتدا هو ".NTX" إلا أنه بإمكانك استخدام ملفات الفهرسة التي تستخدمها (.NDX) dBASE III PLUS والمألوفة لديك بشرط ربط ملف NDX.OBJ الـــذي يأتي مع حزمـــة Clipper ضمـــن براجـــك. إلا أن ملف NTX. أسرع في البحث وتـرتب بيانـات الملف الأصـلي في الملف المفهـرس طبقـا لترتيب ورودها في <key expression> وهـو الحقل المختار لتظهر السجلات طبقا لبياناته. ويمكن أن يكـون <key expression> حقلا أو تعبيرا حرفيا أو رقميا أو تاريخيا. وهذا يعني أن العبارات المنطقية أو حقول الملاحظات (memo fields) لا يمكن أن تستخدم كمفتاح لتفهرس سجلات الملف الأصلي تبعا لمحتوياتها. ويمكن أن يشتمل المفتاح <key expression> على أكثر من حقل مرتبطين بعلامة + بحيث لا يزيد طوله عن ٢٥٠ حرفاً.

ويجب أن تكون الحقول أو العبارات المرتبطة بعلامة + في <key expression> من نفس النوع (type). وفي حالة ربط حقول أو عبارات حرفية مع أخرى رقمية أو تاريخية داخـل <key expression> يمكنـك استخـدام الـوظيفـة ( )STR والـوظيفة

DTOS() للتحويل من حقل رقمي أو تاريخي إلى حرفي. وملف الفهرس ملف متصل بملف قاعدة البيانات الأصلي ويشتمل على رقم السجل الذي تخصصه قاعدة البيانات للسجـلات والمفتاح <key fields> ، ويمكن أن تضع في الذاكرة عددا من ملفات الفهرسة لا يتجاوز ١٥ ويكون أول ملف منها هو الرئيسي (master) والذي يتم البحث فيه في حالة استخدام أحد أوامر البحث SEEK أو FIND .

لحذف السجلات المكررة في الحقل المختار < key fields> عند وضعها على ملف الفهرس. استخدم أمر SET UNIQUE ON قبل عملية الفهرسة.

يشتمل ملف الفهرسة على السجلات المعلمة لفرض الحذف بأمر DELETE والمستبعدة بأمر SET FILTER وعادة يتم ترتيب ملف الفهرسة تصاعدياً فإذا رغبت في الترتيب التنازلي استخدم الوظيفة DESCEND() وفي هذه الحالة استخدم الوظيفة DESCEND() ضمن أمر SEEK أو FIND عندما تلجأ للبحث في الملف.

الاختلاف عن dBASE III PLUS : لا تتعامل dBASE III PLUS مع ملف NTX. وتتعامل مع سبعة ملفات مفهرسة فقط في حين يتعامل Clipper مع ١٥ . لا تستطيع «دي بيس» إجراء الفهرسة تنازلياً ولا على حقول التاريخ بينما يستطيع «كلبر».

أمثـــلة :

(١) لفهـرسـة ملف STOCK.dbf ووضع الترتيب الجديد على ملف مفهرس اسمه IACCOUNT.NTX طبقا لبيانات حقل ACCOUNTNO استخدم الأمر التالي:

```
INDEX ON ACCOUNTNO TO IACCOUNT
```

(٢) ولربط حقل DATE مع حقل ACCOUNTNO

```
INDEX ON ACCOUNTNO + DTOS(DATE) TO ACCDATE
```

(٣) فإذا أردت الحصول على الترتيب بحيث تكون التواريخ تنازلية استخدم الأمر بالصيغة التالية:

```
INDEX ON ACCOUNTNO + DESCED(DTOS(DATE)) TO ACCDATE
```

(٤) وللحصول على قائمة بأسماء الشركات أبجديا من حقل COMPANY

```
LIST ACCOUNTNO,COMPANY
```

(٥) المثال التالي يستخدم تعبير يشتمل على الاسم الأول والأخير بعد حذف الفراغات التي قد توجد بعد الاسم الأول حتى لا تؤثر في نتيجة الفهرسة ولذلك استخدمنا الوظيفة TRIM() ضمن التعبير لحذف الفراغات الموجودة على يمين الاسم الأول ثم أضفنا فراغات بعد الاسمين توازي طول الحقلين. ولأن طول كل حقل = ١٢ فلن نحتاج لأكثر من ٢٤ خانة في عبارة الفرز.

```
INDEX ON SUBSTR(TRIM(FIRSTNAME)+LASTNAME+SPACE(12),1,24)
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

CLOSE – FIND – REINDEX – SEEK – SET – INDEX – SET UNIQUE

الأمر INPUT

يخزن تعبيرا ما داخل حقل ذاكرة.

الشكل العام:

INPUT [<*prompt*>] TO <*memvar*>

حيث:

<*prompt*> : رسالة أو عبارة تظهر على الشاشة أثناء تنفيذ الأمر.

TO <*memver*> : اسم حقل الذاكرة الذي ستوضع داخله المدخلات.

الشرح:

يقبل هذا الأمر قيمة حرفية أو رقمية أو تاريخية أو منطقية تدخل من لوحة المفاتيح وتخزن هذه القيمة داخل حقل ذاكرة (<*memver*>) .

ويتحدد نوع البيانات المخزنة بالذاكرة تبعا لنوع المدخلات فإذا كانت المدخلات تعبيرا حرفيا فإما أن توضع بين علامتي تنصيص "" أو تشتمل على اسم حقل حرفي. وإذا كانت تعبيرا رقميا فإما أن تشتمل على أرقام أو على اسم حقل رقمي . . . . وهكذا.

وتعرف قاعدة البيانات نهاية المدخلات بضغط مفتاح Enter . إذا اشتمل الأمر على الاختيار <*prompt*> فستظهر الرسالة المختارة (<*prompt*>) قبل إدخال أي شيء إلى الذاكرة.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثال:

يوضح المثال الآتي كيف تستخدم هذا الأمر لتخزين تعبير حرفي أو رقمي أو منطقي داخل الذاكرة.

```
INPUT "Enter employee name: " TO M_NAME
```

والاجابة الصحيحة رداً على الرسالة هي : "Mohammed"

```
INPUT "Enter employee salary: " TO M_SALARY
```

والاجابة الصحيحة رداً على الرسالة هي : 4500.50

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

ACCEPT – WAIT – STORE

الأمر JOIN

يربط سجلات وحقول من ملفي قاعدة بيانات ويضع الناتج في ملف قاعدة بيانات جديد.

الشكل العام:

JOIN WITH <*alias*>/(<*expC1*>) TO <*new file*> /(<*expC2*>) FOR <*condition*> [FIELDS <*field list*>]

حيث:

<alias>/(<*expC1*>)	:	اسم بديل للملف قاعدة البيانات الذي سيرتبط مع الملف المفتوح ودائماً يكون اسم المنطقة المختارة.
<new file>/(<*expC2*>)	:	اسم الملف الجديد المزمع إنشاؤه.
FOR <condition>	:	يجعل الأمر يربط فقط السجلات التي تتطابق مع الشرط المحدد في الأمر <condition>.
<field list>	:	أسماء الحقول المختارة مفصولة بعلامة « , ».

الشرح:

يستخدم هذا الأمر لربط ملفي قاعدة بيانات أحدهما يجب أن يكون مفتوحا (Active) والثاني موجودا على القرص (يعرف للأمر بالاختيار <alias>) ويضع الناتج على ملف قاعدة بيانات جديد (يعرف للأمر بالاختيار (<new file>) ويضع هذا الأمر المؤشر على أول سجل في الملف المفتوح ويبحث في السجلات الموجودة في الملف الآخر <alias> عن السجلات التي تتطابق مع الشرط المحدد في الأمر بالاختيار <condition> . وكلما وجد سجلا في الملف الثاني <alias> متطابقا مع الشرط المذكور في <condition> كتبه على الملف الجديد <new file> وهكذا إلى أن تنتهي جميع

السجلات الموجودة في الملف المفتوح. بحيث ينتج في النهاية ملفا جديدا يشتمل على جميع السجلات التي تتطابق مع الشرط المذكور في الأمر مشتملا على جميع الحقول المتوفرة في الملفين. فإذا اشتمل الأمر على الاختيار [FIELDS <field list>] فإن الملف الجديد سيشتمل على الحقول المذكورة بعد كلمة FIELDS فقط.

ويجب استخدام هذا الأمر بحذر شديد لأن عدد السجلات التي يجري تشغيلها على القرص في النهاية يعادل عدد سجلات الملف الأول مضروبة في عدد سجلات الملف الثاني ولذلك يجب أن تكون المساحة المتاحة على القرص كافية لتشغيله.

الاختلاف عن dBASE III PLUS : غير موجود بها.

مثـــال :

ينشيء المثال التالي ملفا جديدا باسم STCOURSE.dbf مشتملا على حقلين من ملف STUDENTS.dbf هما LASTNAME, FIRSTNAME وحقلين من ملف COURSES.dbf هما COURSE1, COURSE2.

```
SELECT B
USE COURSES
SELECT A
USE STUDENTS
JOIN WITH B TO STCOURSE FOR STUDENTNO = B->STUDENTNO FIELDS FIRSTNAME,;
 LASTNAME, B->COURSE1, B->COURSE2
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

APPEND FROM – SET FIELDS – SET RELATION

الأمــر KEYBOARD

يمسح المحطة الانتقالية الموجودة بالذاكرة (Typeahead buffer) ويضع بها عبارة حرفية .

الشكل العام:

KEYBOARD <*expC*>

حيــث:

<*expC*>) : العبارة التي ستوضع في المحطة الانتقالية .

الشـــرح:

العبارة الحرفية التي توضع في الذاكرة في منطقة تسمى **Typeahead buffer** أو المحطة الانتقالية يمكن قراءتها فيما بعد بالأوامر الموجودة في البرنامج كما لو دخلت من لوحة المفاتيح بواسطة المستفيد من النظام . ولذلك فإن استخدام هذا الأمر يكون مفيداً عندما ترغب في جعل اختيارات قائمة النظام تتم تلقائيا بدون تدخل المشغل كما يحدث في البرامج الاستعراضية التي تعدها بعض الشركات . ويتم مسح المحطة الانتقالية في كل مرة ينفذ فيها أمر KEYBOARD .

الاختلاف عن *dBASE III PLUS* : غير موجود بها .

مثـــال:

المثال الآتي يتحكم في اختيار الرقم ١ من القائمة الأولى للنظام تلقائياً والرقم ٢ من القائمة الثانية والحرف (Q) من القائمة الثالثة

```
KEYBOARD "1" + CHR(13) + "2" + CHR(13) + "Q"
```

والمثال التالي يمسح المحطة الانتقالية من أية حروف بها وذلك لأنه يستخدم الأمر بدون إدخال حروف ويفيد هذا المثال إذا أردت أن تتأكد أن المحطة الانتقالية خالية تماماً قبل

أمر READ أو GET لتتأكد أن البرنامج لن يتأثر بأية بيانات غير التي يدخلها المشغل
KEYBOARD CHR(0)

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

CLEAR TYPEAHEAD – LASTKEY( ) – INKEY( )

الأمـــر LABEL

يظهر أو يطبع ملصقات بريدية من ملف ملصقات (LBL.) .

الشكل العام:

LABEL FORM <*labelfile*>/(<*expC1*>) [<*scope*>] [SAMPLE] [TO PRINT]

[FOR <*condition*>] [WHILE <*condition*>] [TO FILE <*file*>/(<*expC2*>)]

حيـــث:

<*lable file*>/(<*expC1*>)	:	اسم الملف الموجود عليه الملصقات.
<*scope*>	:	يحدد السجلات التي ستختار من الملف لتنفذ مع الأمر.
FOR <*condition*>	:	تسمح بتنفيذ الأمر مع السجلات التي تتطابق مع الشرط الموجود بالأمر.
WHILE <*condition*>	:	تسمح بتنفيذ الأمر طالما أن الشرط المذكور بالأمر صحيحا.
<*file*>/(<*expC2*>)	:	اسم الملف الذي ستوضع عليه الملصقات.
SAMPLE	:	يظهر عينة من شكل الملصقة.
TO PRINT	:	تظهر الملصقات على الطابعة أيضا.

الشـــرح:

يستخدم هذا الأمر لاستخراج ملصقات من ملف قاعدة البيانات المفتوح. هذه الملصقات موجودة على ملف سبق إنشاءه، ويشار إليه في الأمر بالعبارة <*labelfile*> . وهـذا الملف يتم إنشـاؤه باستخـدام ملف RL الـذي يأتي مع حزمة Clipper بدون استخدام dBASE III PLUS .

إذا لم يشتمـل الأمـر على أحـد الاختيارات <*scope*> أو FOR/WHILE فإن الملصقات ستظهر من جميع السجلات أما في حالة اختيار أحد هذه الاختيارات فإن

السجلات المتطابقة مع الشرط الموجود في الأمر فقط هي التي ستظهر لها ملصقات.

الاختيار TO PRINT يظهر الملصقات على الطابعة والاختيار TO FILE يرسل الملصقات إلى ملف خارجي على القرص يأخذ اسما ممتدا هو ".txt".

استخدم الاختيار SAMPE لاظهار عينة من الملصقات قبل طباعتها لانها تعطيك الفرصة لاظهار حجم الملصقة ومدى مناسبتها للورق المخصص للطباعة.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

المثال التالي يستخدم ملف STUDENTS.dbf لاظهار محتويات ملف الملصقات (label form) الذي سبق انشاؤه وحفظه من قبل باسم STLBL.lbl وإرسال الناتج إلى الطابعة

```
USE STUDENTS
LABEL FORM STLBL TO PRINT
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

REPORT FORM – SET PRINTER

الأمـــر LIST

يظهر محتويات ملف قاعدة البيانات.

الشكل العام:

LIST [<*scope*>] [<*expression list*>] [FOR <*condition*>]

[WHILE <*condition*>] [OFF] [TO PRINT] [TO FILE <*file*>/(<*expC*>)]

الاختيارات المتـاحـة مع هذا الأمـر هي نفس الاختيارات المتـاحـة مع أمـر DISPLAY الذي سبق شرحه في هذا الفصل وهذا الأمر مشابه تماما لأمر DISPLAY باستثناء واحد وهو:

إذا لم تحدد الاختيار <*scope*> في أمر LIST فإن القيمة التلقائية هي ALL في حين أنها في أمر DISPLAY رقم السجل الذي يقف عنده المؤشر.

الاختلاف عن *dBASE III PLUS* : لا تتعامل مع الاختيار TO FILE .

مثـــال:

المثال الآتي يستخدم ملف STOCK.dbf لطباعة محتويات الملف بدون إظهار رقم السجل (Record #) واختيار السجلات التي تزيد أسعارها (PRICE) عن رقم ١٠٠

```
USE STOCK
LIST TO PRINT OFF FOR PRICE < 100
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET CONSOLE – SET DELETED

الأمر LOCATE

يضع المؤشر عند السجل الذي يتطابق مع الشرط الموجود بالأمر.

الشكل العام:

LOCATE [<*scope*>] [FOR <*condition*>] [WHILE <*condition*>]

حيث:

<*scope*> : تحدد السجلات التي سيتم البحث عنها (مثل all أو rest . . . الخ .

FOR <*condition*> : تسمح بالبحث في السجلات التي تتطابق مع الشرط المذكور في الأمر.

WHILE <*condition*> : تسمح بالبحث طالما أن الشرط الموجود بالأمر صحيحا.

الشرح:

يبحث هذا الأمر داخل ملف قاعدة البيانات عن أول سجل يتطابق مع الشرط المذكور بالأمر. وعادة يتم البحث في الملف كله ابتداء من السجل الأول ما لم يذكر في الأمر الاختيار <*Scope*> . فإذا اشتمل الأمر على الاختيار <*Scope*> فإن السجلات المذكورة فقط هي التي يتم البحث فيها.

استخدم أمر CONTINUE لتستأنف البحث ابتداء من مكان وقوف المؤشر داخل الملف عن السجل التالي والذي يتطابق مع الشرط المذكور في الأمر.

إذا وجدت قاعدة البيانات السجل المطلوب داخل الملف فإن الوظيفة FOUND( ) تصبح صحيحة (.T.) .

أما إذا لم تجد السجل المطلوب فإن المؤشر يوضع في نهاية الملف وتصبح الوظيفة EOF( ) صحيحة (.T.) أما الوظيفة FOUND( ) فتصبح غير صحيحة (.F.) .

وهذا الأمر مثل أمر FIND يحتاج أن تصدر أمر SET EXACT ON لتتم مطابقة الحروف الموجودة في الأمر مع محتويات الحقل بالكامل ليتقرر هل المقارنة صحيحة أم لا؟ لتستأنف البحث الذي بدأه أمر LOCATE استخدم أمر CONTINUE .

الاختلاف عن dBASE III PLUS : يستطيع Clipper أن يبحث في أكثر من منطقة في حالة اختيار أكثر من منطقة بأمر SELECT أما dBASE III PLUS فلا تبحث إلا في منطقة مختارة واحدة.

مثـــال :

المثال التالي يستخدم ملف STOCK.dbf وفيه تلاحظ أن أول أمر LOCATE يبحث عن أول سجل يحتوي على ACCOUNTNO مساويا للرقم 066882 وفي نفس الوقت PRICE أقل من 100 .

```
USE STOCK
LOCATE FOR ACCOUNTNO = "066882" .AND. PRICE < 100
? EOF()                    && .F.  الإجابة:
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

CONTINUE – FOUND( ) – SEEK – FIND

الأمر MENU TO

ينفذ قائمة اختيارات تستخدم الشريط المضاء طبقاً للأوامر التي سبق تحديدها بمجموعة أوامر PROMPT .

الشكل العام:

MENU TO <*memvar*>

حيث:

<*memvar*> : اسم المكان الذي سيتم فيه تخزين اختيارات القائمة.

الشرح:

ينفذ هذا الأمر قائمة تسمح للمستفيد من النظام بالتحرك بين اختياراتها باستخدام الشريط المضاء، ويتم اختيار واحد من اختيارات القائمة بتحريك الشريط المضاء باستخدام مفاتيح الأسهم إلى الاختيار المطلوب ثم الضغط على أحد المفاتيح التالية:
مفتاح Enter أو PgDn أو PgUp أو الضغط على الحرف الأول من الاختيار المطلوب.

ويتطلب هذا الأمر وجود مجموعة من أوامر PROMPT...@ التي تقوم بتعريف اختيارات القائمة وتقتصر مهمته على استدعاء القائمة والسماح بالتحرك بين اختياراتها عن طريق مفاتيح الأسهم. ويقوم هذا الأمر بتخزين الرقم الذي يخص الاختيار الموجود تحت الشريط المضاء في حقل ذاكرة. ولذلك يمكنك استخدام هذا الرقم للتفريع باستخدام أمر DO CASE . فإذا ضغط المستفيد مفتاح Esc فيتم تخزين الرقم صفر في حقل الذاكرة.

إذا أردت الانتقال حول الاختيارات كلها، بعبارة أخرى، إذا أردت الانتقال إلى أول اختيار عندما تصل إلى آخر اختيار تلقائيا استخدم أمر

SET WRAP ON

والحد الأقصى من الاختيارات المسموح به داخل قائمة واحدة هو ٣٢ اختياراً. وأثناء وضع الشريط المضاء على أحد الاختيارات فإن الرسالة المحددة في الأمر PROMPT...@ تظهر في السطر المحدد بالأمر SET MESSAGE .

ويوضح الجدول التالي وظائف بعض المفاتيح أثناء ظهور القائمة:

المفتاح	وظيفتـــه
↑	الاختيار السابق.
↓	الاختيار اللاحق.
Home	أول اختيار في القائمة.
End	آخر اختيار في القائمة.
←	الاختيار السابق.
→	الاختيار اللاحق.
PgUp	اختيار الاختيار الحالي.
PgDn	اختيار الاختيار الحالي.
Enter	اختيار الاختيار الحالي.
Esc	الخروج من القائمة.

الاختلاف عن dBASE III PLUS : غير موجود بها.

مثـــال :

المثال التالي يوضح القائمة الرئيسية لنظام الطلاب وهي تشتمل على أربعة اختيارات. وفي هذا المثال عندما يضغط المستفيد مفتاح الادخال أثناء وضع الشريط المضاء على أحد الاختيارات فإن رقم هذا الاختيار يخزن في حقل الذاكرة ACTION . فمثلاً الاختيار الأول يضع الرقم ١ في حقل الذاكرة ACTION والاختيار الثاني يضع الرقم ٢ . . . وهكذا. أما إذا ضغط المستفيد مفتاح Esc فسيوضع الرقم صفر في الحقل

ACTION . أما أمر DO CASE في هذا المثال فمهمته تقييم الاختيار الذي تم، واستدعاء البرنامج المناسب.

```
SET WRAP ON          && اسمح للمؤشر بالانتقال لأول اختيار اذا وصل للأخير
SET MESSGAE TO 1     && اظهر الرسالة الاختيارية فى السطر رقم 1
@ 10,05 PROMPT " Maintenance    " MESSAGE " Add, Delete, Edit    "
@ 12,05 PROMPT " Query          " MESSAGE " Ask questions        "
@ 14,05 PROMPT " Reports        " MESSAGE " Reports Menu         "
@ 16,05 PROMPT " Exit           " MESSAGE " Exit to DOS          "
MENU TO ACTION
DO CASE
  CASE ACTION = 1         && اذا اختير أول اختيار
    DO STMAINT
  CASE ACTION = 2         && اذا اختير ثانى اختيار
    DO STINQ
  CASE ACTION = 3         && اذا اختير ثالث اختيار
    DO STRPT
  CASE ACTION = 4 .OR. ACTION = 0  && Esc اذا اختير رابع اختيار أو مفتاح
     RETURN
ENDCASE
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@...PROPT – SET MESSAGE – SET WRAP – ACHOICE( )

الأمـــر NOTE/*/&&

يظهر أمام سطر الملاحظات داخل البرنامج .

الشكل العام:

يأخذ الشكل العام للأمر إحدى الصيغ الثلاثة الآتية :

NOTE <*text*>

* <*text*>

<*command*> && <*text*>

حيـــث:

<*text*> : نص التعليق أو الملاحظة المطلوب إدراجها بالبرنامج .

الشـــرح:

إذا أردت أن تكتب تعليقات أو ملاحظات داخل البرنامج فيمكنك ذلك بإحدى طريقتين :

١ ـ أن تبدأ السطر بكلمة NOTE أو علامة * ثم تتبعها بالملاحظة أو التعليق الذي تريده، وفي هذه الحالة فإن قاعدة البيانات لن تنفذ هذا الأمر ولن تطبق عليه قواعد اللغة .

٢ ـ أن تكتب الأمر المطلوب للتنفيذ داخل البرنامج وتترك مسافة خالية على الأقل ثم تتبعها بعلامتي && ثم تكتب الملاحظة (<*text*>) التي تريدها .

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف .

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة : لا توجـــد .

الأمر PACK

يحذف السجلات المعلمة بعلامة الحذف نهائيا من الملف.

الشكل العام:

PACK

الشرح:

يحذف هذا الأمر السجلات المعلمة لأغراض الحذف من ملف قاعدة البيانات المفتوح حذفا نهائيا. ولا يمكن استرجاعها مرة ثانية من الملف. وتتعدل سجلات ملفات الفهرسة المفتوحة تلقائيا بعد تنفيذ هذا الأمر بناءً على الوضع الجديد للملفات.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثال:

المثال التالي يحذف كل السجلات المعلمة لأغراض الحذف من ملف STUDENTS.dbf ويعيد فهرسة ملف ILAST.NTX تلقائيا.

٧ - ٢

```
USE STUDENTS INDEX ILAST
PACK
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة:

DELETE - DELETED ( ) - SET DELETED - ZAP - REINDEX - RECALL

الأمـــر PARAMETERS

يعرف المعطيات التي ستستخدم مع أمر DO...WITH .

الشكل العام:

PARAMETERS <*Parameter List*>

حيـــث :

<*Parameter List*> : أسماء لحقول الذاكرة مفصولة بعلامة « , » .

الشـــرح :

يخصص هذا الأمر أسماء لحقول الذاكرة (Memvar Variables) هذه الأسماء هي التي تذكر بعد أمر PARAMETERS ويفصل بين كل اسم وآخر بعلامة « , » ويتم إدخال محتويات حقول الذاكرة من برنامج آخر يسمى Calling Program وتدخل بعد الاختيار WITH في أمر DO الذي يستدعي البرنامج (Called Program) أو الاجراء الـذي يشتمـل على أسـماء حقـول الذاكرة أو بعبارة أخرى الذي يشتمل على قائمة المعطيات الموجودة في أمر PARAMETERS .

ويجب أن يكـون أمـر PARAMETERS هو أول أمـر في البرنامج أو الاجراء المستدعى (Called) . ولا يشترط أن تكون الأسماء الموجودة بالأمر مساوية في عددها لعدد المعطيات التي تدخل في أمر DO...WITH . فإذا أردت أن تعرف عدد المعطيات استخدم الوظيفة PCOUNT() .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف

أمثـــلة :

(١) يشتمل المثال التالي على أمر PARAMETER في أبسط صورة

```
* Program: AREA.PRG
PROCEDURE RECT
PARAMETERS LENGTH,WIDTH,AREA
AREA = LENGTH * WIDTH
RETURN
```

ولتنفيذ هذا الاجراء أدخل الأمر التالي:

```
DO RECT WITH 8,5,0
? AREA                          && 40 اجابة كلبر:
```

(٢) عادة يتم قبول المعطيات بقيمها للتعويض عنها في الوظائف الخاصة (User Defined Function) . إلا أنه يجوز قبول المعطيات بالاشارة إلى اسم الحقل الذي يشتمل عليها إذا سُبق اسم الحقل بعلامة @ . وللتوضيح نسوق المثال التالي:

```
* Program: PRICE.PRG
PRICE = 4500
? NEWPRICE (@PRICE)
? PRICE                          && 5220 اجابة كلبر:
*                                && 4500 اجابة كلبر:
FUNCTION NEWPRICE
  PARAMETERS NPRICE
  NPRICE = NPRICE * 1.15
RETURN NPRICE
```

وفي هذا المثال فإن الوظيفة NEWPRICE() تحسب الأسعار الجديدة بعد زيادة الأسعار بنسبة ١٥٪. وفي هذه الوظيفة فإن القيمة PRICE تم تعريفها للوظيفة بالاشارة إلى اسم الحقل الذي يشتمل عليها وليس بالسعر نفسه. ولذلك فإن أي تغيير في قيمة NPRICE الموجودة بداخل الوظيفة سيؤثر على PRICE .

(٣) ويمكن تعريف معطيات حرفية من خلال DOS بحيث يفصل بين كل منها فراغ بالشكل الآتي:

C:\> <*programname*> P1 P2 P3

فمثلا إذا أردت تنفيذ برنامج اسمه SALE.EXE بحيث يقبل اسم الصنف (PARTNAME) ورقمه (PARTNO) من محث DOS أدخل الأمر الآتي:

C:\ SALE "Women Clothes" 114

وحتى لا يلتبس الأمر على «كلبر» في هذا المثال ولكي يفهم أن (Women clothes) متغير واحد استخدمنا علامات التنصيص حول اسم الصنف لأنه يشتمل على فراغ، والمفروض أن الفراغ فاصل بين متغير (Parameter) وآخر.

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

DO – PRIVATE – PUBLIC – SET PROCEDURE – PCOUNT( )

الأمر PRIVATE

يخصص حقول للذاكرة أو مصفوفات تستخدم فقط داخل البرنامج الذي ينشئها.

الشكل العام:

PRIVATE *<memvar list>* / *<array list>*

حيث:

<memvar list> : أسماء حقول الذاكرة *<memory variables>* مفصولة بعلامة « , ».

<array list> : اسم مصفوفة أو مصفوفات مفصولة بعلامة « , ».

الشرح:

يستخدم هذا الأمر داخل البرنامج ليعلن أن حقول الذاكرة المذكورة أسماءها مخصصة لهذا البرنامج فقط.

بمعنى أن نفس الأسماء إذا خصصت لحقول ذاكرة (Memvar) مع برامج في مستويات أخرى غير المستوى الذي أنشئت بداخله فلن تتأثر محتويات الذاكرة التي أنشئت بأمر PRIVATE وستبقى قيمتها كما هي. وذلك لأن حقول الذاكرة الخاصة يتم إلغاؤها قبل انتهاء البرنامج والعودة إلى البرنامج الذي استدعاه.

الاختلاف عن dBASE III PLUS : لا تتعامل «دي بيس» مع المصفوفات ولا يتعامل «كلبر» مع الاختيارات ALL وLIKE وEXCEPT الموجودة في الأمر.

مثال:

المثال التالي عبارة عن برنامج رئيس يستدعي برنامجاً فرعيا وقد تم تخصيص نفس حقول الذاكرة في البرنامج الفرعي SUBPROG.PRG على أنها PRIVATE ولذلك عندما رجع التنفيذ إلى البرنامج الرئيس MASTER.PRG ظهرت محتويات

الحقول المعرفة من قِبَلِه وليست الأخيرة في الذاكرة التي أنشئت داخل برنامج SUB-PROG.PRG .

```
* Program: MASTER.PRG
ONE = 111
TWO = " A test program"
DO SUBPROG
? " First:" ,ONE              && First: 111 اجابة كليبر:
? " Second: ",TWO             && Second: A test program اجابة كليبر:
RETURN
*
* PROGRAM: SUBPROG.PRG
PRIVATE ONE,TWO
ONE = 999
TWO = "Don't change me"
? "First: ",ONE               && First: 999 اجابة كليبر:
? "Second: ",TWO              && Second: Don't change me اجابة كليبر:
RETURN
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

PARAMETERS – PUBLIC

الأمر PROCEDURE

يعلن بداية برنامج صغير يسمى Procedure يتم تنفيذه عند استدعائه .

الشكل العام:

PROCEDURE <*procedure name*>

<*Commands*>

[RETURN]

حيث:

<*procedure name*> : اسم البرنامج الصغير. ويسمى إجراء أو Procedure .

الشرح:

يستخدم هذا الأمر ليعرِّف بداية برنامج صغير يسمى procedure ، ويتم وضعه في أي مكان داخل ملف البرنامج .

وطبعا يوفر هذا المفهوم وضع كل برنامج في ملف مستقل وبالتالي يوفر المساحة المستخدمة على القرص .

ويتعامل Clipper مع الاجراءات بطريقة مختلفة dBASE III PLUS فلأن كل الأوامر الموجودة في البرنامج يتم ترجمتها ووضعها في الذاكرة أثناء التنفيذ فليس هناك ضرورة لفتح أو غلق ملف الاجراءات ، ولذلك فإن وضع الاجراء في أي مكان داخل البرنامج لا يسبب مشكلة لأنه عندما يُترجم ضمن البرنامج يصبح متاحاً لكل أجزاء البرنامج ، وأيضاً أمر RETURN في نهاية الاجراء غير ضروري لأن «كلبر» يترجم كل الأوامر التي تلي أمر PROCEDURE حتى يصل إلى أمر PROCEDURE جديد .

الاختلاف عن dBASE III PLUS : ليس من الضروري بالنسبة «لكلبر» أن توضع الاجراءات داخل ملف برنامج مستقل (.PRG) ولذلك فإن أمر CLOSE PROCEDURE الذي تستخدمه «دي بيس» غير ضروري مع «كلبر» .

مثـــال:

يشتمل المثال التالي على برنامج رئيس يستدعي برنامجين صغيرين يستخدمان لتغيير ألوان الشاشة موجودان في نفس ملف البرنامج.

```
* Program: PROC1.PRG
DO RED              && RED :انقل التنفيذ الى الاجراء
DO GREEN            && GREEN :انقل التنفيذ الى الاجراء
RETURN
*
PROCEDURE RED                 && Bgeining of the procedure
SET COLOR TO R+/N,GR+/R+
CLEAR
RETURN                        && يمكن اغفال هذا الامر
*
PROCEDURE GREEN               && بداية الاجراء:
SET COLOR TO G+/N,W+/B
CLEAR
RETURN                        && يمكن اغفال هذا الامر
* EOF: PROC1.PRG
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

DO – SET PROCEDURE TO

الأمر PUBLIC

يسمح باستخدام حقول الذاكرة (Memory Variables) والمصفوفات في جميع البرامج وفي كل المستويات.

الشكل العام:

PUBLIC <*memvar list*>/ <*array list*>

حيث:

<*memvar list*> : أسماء حقول الذاكرة <*Memory Variables*> مفصولة بعلامة « , ».

<*array list*> : أسماء المصفوفات (Arrays) مفصولة بعلامة « , ».

الشرح:

يسمح هذا الأمر بالتعامل داخل المستوى الأعلى من البرامج مع البيانات (Memory Variables/Arrays) التي أنشئت داخل المستوى الأدنى وهو يوضع في أي مستوى من البرامج. ويتم التعامل معه من خلال البرنامج الرئيسي أو البرامج الفرعية المتصلة به في كل المستويات.

وبدون استخدام هذا الأمر وحسب نظام قاعدة البيانات فإن المستوى الأعلى من البرامج لا يستطيع التعامل مع البيانات (Memory Variables/Arrays) التي أنشئت داخل المستوى الأدنى.

فإذا أردت أن تجعل قيمة ما (Variable) متاحة لجميع البرامج في جميع المستويات فيجب أن تعلن أنها عامة PUBLIC قبل استخدامها.

الاختلاف عن *dBASE III PLUS* : لا تتعامل «دي بيس» مع المصفوفات.

مثــــال :

في هذا المثـال فإن كلا من المتغـيرين NAME و MSG عام (PUBLIC) لكل البرامج في كل المستويات التي تستخدمها. وكذلك المصفوفة PARRAY

```
PUBLIC MSG,NAME          && هذه المتغيرات عامة لجميع البرامج داخل النظام
MSG = "Invalid selection"
NAME = "Mohammad"
PUBLIC PARRAY [5]
DECLARE PARRAY [5]
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة:

PRIVATE – PARAMETERS – DO

الأمـر QUIT/ CANCEL

يغلق جميع الملفات المفتوحة ويسمح بالخروج من قاعدة البيانات.

الشكل العام:

QUIT

CANCEL

الشـــرح:

يتسبب أحد الأمرين في:

- إغلاق جميع الملفات المفتوحة.
- محو جميع حقول الذاكرة.
- إنهاء العمل مع قاعدة البيانات والعودة إلى نظام التشغيل.

ويجب التنبيه إلى أن إغلاق الحاسب بدون استخدام هذا الأمر قد يتسبب في تخريب بعض الملفات المفتوحة.

الاختلاف عن *dBASE III PLUS*: في «دي بيس» أمر CANCEL ينقل التنفيذ إلى نقطة توجيه الأوامر وليس إلى نظام التشغيل.

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

RETURN

الأمر READ

يسمح بإدخال بيانات إلى الحقول المحددة في أمر GET...@ .

الشكل العام:

READ [SAVE]

حيث:

SAVE : تحفظ الحقول المذكورة بعد أمر GET في الذاكرة بعد إدخال بياناتها.

الشرح:

يسمح هذا الأمر بإدخال أو تعديل بيانات إما إلى حقول داخل الملف المفتوح أو إلى حقول ذاكرة (Meomry Variables) سبق تعريفها. هذه الحقول تظهر على الشاشة بناءً على أمر GET...@ ويظهر المؤشر عند بداية أول حقل تم تعريفه داخل مجموعة الأوامر التي تنتمي إلى GET...@ . ويسمح للمستخدم بإدخال أو تعديل بيانات إلى الحقول الظاهرة على الشاشة. وتتعدل محتويات حقول الملف أو الذاكرة التي ذكرت في أوامر GET...@ بالقيم التي أدخلت إليها.

إذا أصدر أمر READ دون أن تكون أوامر GET...@ قد سبقته فإن البرنامج يتوقف مؤقتا في حالة انتظار حتى يتم ضغط أحد مفاتيح لوحة المفاتيح.

إذا اشتمل الأمر على الاختيار SAVE فإنه يسمح بإظهار الحقول المذكورة في مجموعة أوامر GET...@ لتتعدل بياناتها مرة ثانية عند أول مرة يصدر فيها أمر READ .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثــال:

المثال التالي عبارة عن جزء من برنامج يخصص بعض حقول للذاكرة ويظهرها على الشاشة لادخال بيانات إليها.

```
STORE SPACE(14) TO MLNAME,MFNAME
STORE 0 TO MAGE
@ 10,10 SAY "Last name : " GET MLNAME
@ 12,10 SAY "First name: " GET MFNAME
@ 14,10 SAY "Age       : " GET MAGE
READ
```

في هذا المثال يضع الأمر READ المؤشر عند أول حقل في مجموعة أوامر GET...@ ألا وهو MLNAME ويسمح بتعديله.

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

CLEAR GETS – @...GET – SET FORMATE TO

الأمـر RECALL

يسترجع سجلات علمت لغرض حذفها.

الشكل العام:

RECALL [<*scope*>] [FOR <*condition*>] [WHILE <*condition*>]

حيـــث:

<scope>	:	يحدد السجلات التي سيتم استرجاعها من الملف (مثل all أو rest . . . . . الخ).
FOR <condition>	:	يسترجع كل السجلات التي تتطابق مع الشرط الموجود في الأمر.
WHILE <condition>	:	تسمح بتنفيذ الأمر طالما أن الشرط الموجود بالأمر صحيحا.

الشـــرح:

يستخـدم هذا الأمر لالغاء علامة الحذف التي وضعت أمام السجلات. أي لاعادة السجلات التي تم تعليمها بالأمر DELETE إلى حالتها السابقة.

وينفذ الأمر مع السجل الذي يقف عنده المؤشر ما لم يستخدم الاختيار <scope> أو FOR/WHILE <condition> . فإذا استخدم أحد هذه الاختيارات فإن جميع السجلات المعلمة لأغراض الحذف والمتطابقة مع الشرط الموجود في الأمر ترجع إلى حالتها السابقة. ولن يتم حذفها فيها بعد عند إصدار أمر PACK .

إذا كان أمر SET DELETED في حالة ON فإن الأمر سيسترجع سجلاً واحداً فقط وهو السجل الذي يقف عنده المؤشر أو السجل الذي يذكر رقمه.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

الأوامر التالية توضح لك تأثير أمر RECALL .

```
USE STUDENTS
DELETE RECORD 5
?DELETED()                 && .T. اجابة كليبر:
RECALL
?DELETED()                 && .F. اجابة كليبر:
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

DELETE – DELETED ( ) – PACK – SET DELETED – ZAP

الأمـــر REINDEX

يعيد فهرسة ملف قاعدة البيانات في حالة وجود ملف فهرس.

الشكل العام:

REINDEX

الشـــرح:

يستخدم هذا الأمر لاعادة بناء ملف الفهرسة في حالة إدخال سجلات إلى ملف قاعدة البيانات بدون فتح ملف الفهرسة. ويجب أن تفتح الملف أو الملفات التي تريد إعادة فهرستها قبل إصدار هذا الأمر لأنه يؤثر فقط على الملفات المفتوحة. ويتم فتح ملف الفهرس إما باستخدام الاختيار INDEXمع أمر USEأو بإصدار أمر SET IN- DEX TO .

الملفات التي سبق فهرستها باستخدام الاختيار UNIQUE مع أمر INDEX يعاد فهرستها بنفس الصيغة أي مع حذف السجلات المكررة من ملف الفهرس.

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثـــال:

إذا احتجت لاعادة فهرسة ملفي ICOMP.NTX و IACC.NTX المتصلين مع ملف قاعدة البيانات STOCK.dbf يجب إدخال الأوامر التالية:

```
USE STOCK
SET INDEX TO ICOPM,IACCOUNT
REINDEX                    && ICOMP.NTX و IACCOUNT.NTX إعادة فهرسة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

INDEX – PACK – SET INDEX – SET UNIQUE – USE

الأمر RELEASE

يلغي محتويات الذاكرة المحددة بالأمر.

الشكل العام:

RELEASE <*memvar list*>

RELEASE ALL [LIKE/EXCEPT <*skeleton*>]

حيث:

<*memvar list*> : يحدد أسماء حقول الذاكرة المطلوب إلغائها.

<*skeleton*> : علامات تستخدم لتعريف حقول الذاكرة المتشابهة.

الشرح:

* إذا اشتمل الأمر على الاختيار <*memver list*> فإن حقول الذاكرة المسماة فقط هي التي تلغى.
* وإذا اشتمل على الاختيار ALL فإن جميع حقول الذاكرة تلغى ما لم يتم تحديد اختيار آخر وهو LIKE أو EXCEPT . وإليك معنى كل من LIKE وEXCEPT .

تعني <*skeleton*> LIKE الغاء حقول الذاكرة التي تشتمل على نفس الحروف الموجودة بالاختيار <*skeleton*> فقط. وتعنى <*skeleton*> EXCEPT إلغاء جميع حقول الذاكرة ماعدا تلك التي تشتمل على حروف مشابهة للحروف الموجودة بالاختيار <*skeleton*> وتستبدل <*skeleton*> بأحد الرمزين * أو ؟.

ويستخدم الرمز * للتعويض عن غياب مجموعة حروف في حين يستخدم الرمز ؟ للتعويض عن غياب حرف واحد.

الاختلاف عن *dBASE III PLUS* : تشتمل «دي بيس» على أمر RELEASE MODULE لحذف ملف BIN. أما «كلبر» فلا يحتاج لذلك.

أمثلـــة :

١ ـ يحذف الأمر التالي فقط حقول الذاكرة التي تبدأ بالحرفين db وتشتمل على أي حروف أخرى.

```
RELEASE ALL LIKE db*
```

٢ ـ لحذف حقلي الذاكرة M_NAME و M_NO فقط من الذاكرة استخدم الأمر التالي:

```
RELEASE M_NAME, M_NO
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

CLEAR MEMORY – RESTORE – RETURN – SAVE – STORE

الأمر RENAME

يغير اسم أي ملف.

الشكل العام:

RENAME ***<current file. extension>*** **TO** ***<new file. extension>***

حيث:

<current file. extension> : اسم الملف المطلوب تغيير اسمه.

<new file. extension> : الاسم الجديد للملف.

الشرح:

يستخدم هذا الأمر لتغيير اسم ملف ما من الاسم الموجود به على القرص إلى اسم آخر جديد. ويجب إدراج الاسم الممتد (extension) لكل من اسم الملف القديم <current file.extension> واسم الملف الجديد <new file.extension> . وإذا كان ملف DBF. يحتوي على حقل ملاحظات (Memo) فلا تنسى تغيير اسم ملف DBT. أيضا.

إذا كان الملف موجودا على قرص أو دليل آخر فيجب إدراج اسم القرص أو الدليل قبل اسم الملف (سواء الاسم القديم أو الاسم الجديد).

يجب أن تتأكد أن الملف بالاسم الأول موجودا وأن الاسم الجديد للملف غير موجود على القرص أو الدليل المخصص معك كما يجب الانتباه إلى ضرورة إغلاق الملف قبل تغيير اسمه.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثال:

بفرض أنك تعمل مع مشغل الوحدة C وأن ملفا بالاسم STUDENTS.prg

موجودا على مشغل الوحدة A وأنك تريد تغيير اسمه ليصبح MEMO.txt فيجب إدخال الأمر التالي :

RENAME A:STUDENTS.prg TO MEMO.txt

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

CLOSE – USE

الأمر REPLACE

يستبدل محتويات الحقول المذكورة في الأمر بالقيم المذكورة.

الشكل العام:

REPLACE [<*Scope*>] [<*aliss*>] <*filed*> WITH
<*expression*> [, <*field2*>, WITH <*expression2*>...]
[FOR <*Condition*>] [WHILE <*Condition*>]

حيث:

<scope>	:	يحدد السجلات التي سيتم تعديلها.
<aliss>	:	اسم بديل للمنطقة المختارة مثل: B->STOCK
<field>	:	اسم الحقل الذي ستستبدل بياناته.
<expression>	:	العبارة التي ستحل محل الحقل المراد استبداله (<fields>) .
<field2><exp2>	:	تستخدم في حالة الحاجة إلى استبدال أكثر من حقل بعبارات جديدة.
FOR <condition>	:	يسمح بتعديل السجلات التي تتطابق مع الشرط الموجود في الأمر.
WHILE <condition>	:	يسمح بالتعديل في السجلات التي تتطابق مع الشرط الموجود في الأمر طالما أن الشرط صحيحا.

الشـــرح:

يسمح هذا الأمر باستبدال محتويات حقل ما بقيمة أو بعبارة أخرى تحدد داخل الأمر نفسه. ويتم تعديل الحقل الموجود في السجل الذي يقف عنده المؤشر داخل ملف قاعدة البيانات.

فإذا اشتمل الأمر على الاختيار <Scope> أو FOR/WHILE <Condition> فإن السجلات التي تتطابق مع الحالة المشروحة في الأمر تتعدل جميعها بالقيمة أو العبارة

التي يتضمنها الأمر (<expression>) .

وعـادة تتعـدل بيانـات ملف الفهـرس (Index) المفتوح كلما تعدلت محتويات السجـلات بحيث يوضـع كل سجل في مكانه الصحيح داخل الملف المفهرس[1]. ولذلك يجب أن تكون حذرا عند استخدام هذا الأمر فلا تستخدم الاختيارات (<Scope>) أو FOR/WHILE<Condition> إذا كان الحقل الذي ستستبدل بياناته هو الحقل المستخدم كمفتاح (Index Key) للفهرس.

تستطيع «كلبر» استبدال حقل الملاحظات (Memo) شأنه شأن الحقول الأخرى.

الاختــلاف عن dBASE III PLUS : لا تستــطيع «دي بيس» استبـدال حقـل الملاحظات (Memo) .

مثـــال :

بفـرض أن أسعـار شركة أي . بي . إم زادت بمقدار ٢٠٪ وتريد أن تستبدل محتويات حقل PRICE مع السجلات التي تشتمل في حقل COMPANY على كلمة IBM فقط بالقيمة الجديدة. استخدم الأمر الآتي:

```
REPLACE PRICE WITH PRICE * 1.20 FOR COMPANY = "IBM"
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

EDIT – READ – APPEND BLANK

(١) ويترتب على ذلك أن ينتقل السجل إلى مكانه الجديد داخل الملف.

الأمـر REPORT

يظهر تقريرا سبق تصميمه موجود على ملف FRM .

الشكل العام:

REPORT FORM <*report file*>/(<*expC*> [<*scope*>] [FOR <*condition*>]
WHILE <*condition*>] [PLAIN] [HEADING <*expC*>]
[NOEJECT] [TO PRINT] [TO FILE <*file*>]/(<*expC*>) [SUMMARY]

حيـــث:

<*report file*>/(<*expC*>)	:	اسم ملف التقارير.
<*scope*>	:	تحدد السجلات التي ستظهر ضمن التقرير (مثل REST, ALL . . . الخ).
FOR <*condition*>	:	تظهـر السجـلات التي تتـوافق مع الشرط المذكور بالأمر فقط ضمن محتويات التقرير.
WHILE <*condition*>	:	تسمح باستمرار تنفيذ الأمر طالما أن الشرط الموجود بالأمر صحيحا.
<expC>	:	أي عبارة حرفية.
<*file*>	:	اسم ملف خارجي يوضع عليه التقرير في شكله النهائي.

باقي اختيارات الأمر سيأتي شرحها في الفقرة التالية.

الشـــرح:

يستخـدم هذا الأمـر لاستخـراج تقرير سبق إعداده وحفظه على ملف تقرير (Report File) . وهذا الملف يتم إنشاؤه باستخدام ملف RL الذي يأتي مع حزمة Clip-per .

الاختيار PLAIN :

يمنع إظهار رقم الصفحة والتاريخ التي تظهر عادة في بداية كل صفحة ويظهر العنوان الرئيسي للتقرير في الصفحة الأولى منه فقط.

الاختيار HEADING :

يظهر العنوان المذكور في بداية صفحات التقرير.

الاختيار NOEJECT :

يبدأ طباعة التقرير بدون قفز صفحة قبل عملية الطباعة.

الاختيار TO PRINT :

يظهر التقرير على الطابعة والشاشة. وإذا أردت أن تمنع إظهاره على الشاشة أثناء الطباعة استخدم أمر SET CONSOLE OFF .

الاختيار TO FILE :

يحفظ التقرير على ملف نصي ".txt" . بالشكل الذي يظهر به على الشاشة.

الاختيار SUMMARY :

يظهر فقط الاجماليات الفرعية (Subtotals) والاجمالي النهائي Totals .

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

المثال الآتي يستخدم ملف قاعدة بيانات STOCK.dbf وملف التقرير STKRPRT.frm مع ملاحظة أن ملف قاعدة البيانات تمت فهرسته باستخدام حقل COMPANY على ملف مفهرس اسمه ICOMP .

لطباعة تقرير يشتمل على بيانات شركة IBM فقط استخدم الأوامر التالية :

```
USE STOCK
REPORT FORM STKRPRT FOR COMPANY = "IBM" TO PRINT
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET CONSOLE

الأمر RESTORE

يسترجع حقول ذاكرة سبق تخزينها على ملف خارجي على القرص الممغنط.

الشكل العام:

RESTORE FROM *<memory file>*/(*<expC>*) [ADDITIVE]

حيث:

<memory file>/(*<expC>*) : اسم الملف الموجود عليه حقول الذاكرة.

الشرح:

يسترجع هذا الأمر حقول ذاكرة تم تخزينها على ملف خارجي (Memory File) باستخدام أمر SAVE وعادة يأخذ الملف الذي توضع عليه حقول الذاكرة اسما ممتدا هو (.mem) وعادة تحل حقول الذاكرة التي تم استرجاعها محل الحقول التي كانت موجودة من قبل ما لم يشتمل الأمر على الاختيار ADDITIVE . فإذا اشتمل الأمر على الاختيار ADDITIVE فإن حقول الذاكرة التي سترجع ستضاف إلى الحقول الموجودة من قبل ولن تحل محلها. فإذا تشابهت أسماء حقول الذاكرة الموجودة من قبل مع تلك التي تم استرجاعها فإن الجديدة ستحل محل القديمة.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

أمثلة:

١ - لالغاء جميع حقول الذاكرة وإحلال غيرها التي سبق وضعها على ملف اسمه STMEM.mem محلها استخدم الأمر التالي:

```
RESTORE FROM STMEM
```

٢ - ولاسترجاع حقول الذاكرة التي سبق وضعها على ملف اسمه STMEM.mem بدون إلغاء تلك الموجودة بالذاكرة استخدم الأمر التالي:

```
RESTORE FROM STMEM ADDITIVE
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

PUBLIC – SCREEN – SAVE – STORE – SAVE SCREEN – RESTORE

الأمر RESTORE SCREEN

يسترجع شاشة سبق حفظها.

الشكل العام:

RESTORE SCREEN [FROM <*memvar*>]

حيث:

FROM <*memvar*> : اسم حقل الذاكرة الذي حفظت به الشاشة.

الشرح:

يمكن استرجاع الشاشة التي حفظت بأمر SAVE فإذا كانت حفظت بالذاكرة فيكفي إصدار أمر RESTORE SCREEN وإذا كانت حفظت داخل حقل ذاكرة فلابد من ذكر اسمه عند استرجاعها.

الاختلاف عن *dBASE III PLUS* : غير موجود بها.

مثال:

المثال التالي يسترجع شاشة حفظت بالذاكرة

```
RESTORE SCREEN
```

بينما يسترجع المثال التالي شاشة حفظت تحت اسم: SCRVAR

```
RESTORE SCREEN FROM SCRVAR
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

RESTORE – SAVE SCREEN – SAVE TO

الأمر RETURN

ينهي البرنامج ويعيد التنفيذ إلى البرنامج الرئيسي أو إلى نظام التشغيل.

الشكل العام:

RETURN [<*expC*>]

حيث:

<*expC*> : قيمة تخصص للوظيفة الخاصة إذا كان الأمر في نهاية وظيفة خاصة.

الشرح:

يتسبب هذا الأمر في إنهاء البرنامج أو الاجراء الذي ينفذ ويسمح بالعودة إلى البرنامج الرئيسي أو إلى نظام التشغيل ويلغي حقول الذاكرة (Memory Variables) التي أنشأها هذا البرنامج.

وجود هذا الأمر للدلالة على نهاية الوظيفة الخاصة ضروري. في حين أن وجوده للدلالة على نهاية البرنامج أو الاجراء في ملف البرنامج اختياري.

لا يمكن استخدام هذا الأمر بصيغة RETURN TO MASTER التي توفرها dBASE III PLUS

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» الاختيار (<*expC*>).

مثـال:

المثال الآتي يستخدم الأمر RETURN لاعادة التنفيذ للبرنامج الرئيسي في حالة اختيار "N" ردا على رسالة الانتظار:

```
WAIT "Are you sure your printer is ready? [Y/N]" TO YN
IF UPPER(YN) = "N"
  RETURN
ELSE
  * <Commands>
ENDIF
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

DO – PROCEDUR – QUIT

الأمر RUN أو !

يشغل برنامج خارجي أو أحد أوامر نظام التشغيل من خلال قاعدة البيانات.

الشكل العام:

RUN *<commands>* / (*<expC>*)

! *<commands>* / (*<expC>*)

حيث:

<commands>/<expC> : أحد أوامر نظام التشغيل المطلوب تنفيذه.

الشرح:

كل من RUN أو ! يقوم بنفس العمل وهو استدعاء أحد أوامر نظام التشغيل أو برنامج خارجي لتنفيذه وبعد انتهاء التنفيذ يتم الرجوع إلى البرنامج مرة أخرى.

ويجب أن تتأكد أن مساحة الذاكرة تكفي لتشغيل البرنامج الخارجي بالاضافة إلى المساحة المطلوبة لتنفيذ أمر RUN وحده. فيحتاج أمر RUN لمساحة قدرها ١٧ ك.ب بالاضافة إلى المساحة التي يشغلها البرنامج داخل ذاكرة الحاسب.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثال:

لكي تستدعي برنامج اسمه OUT.EXE للتنفيذ أدخل الأمر بإحدى الصيغتين التاليتين:

```
RUN OUT
! OUT
```

وعندما ينتهي تنفيذ برنامج OUT سيرجع التنفيذ إلى الأمر التالي لأمر RUN داخل النظام.

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة:

DO

الأمر SAVE

يحفظ محتويات الذاكرة على ملف خارجي .

الشكل العام:

SAVE TO <*file*>/(<*expC*>) [ALL LIKE/EXCEPT <*skeleton*>]

حيث :

file/<*expC*>	:	اسم الملف الذي ستوضع عليه محتويات الذاكرة.
ALL LIKE <skeleton>	:	يخزن كل محتويات الذاكرة التي توافق سمتا معينا (skeleton) .
ALL EXCEPT <skeleton>	:	يخزن كل محتويات الذاكرة التي لا توافق سمتا معينا (skeleton) .
<skeleton>	:	إما علامة * وتعني غياب مجموعة أحرف أو علامة ? وتعني غياب حرف واحد.

الشرح :

يحفظ هذا الأمر محتويات الذاكرة (Memory Variables) على ملف خارجي على القرص ليمكن استرجاعها مرة ثانية بالشكل الذي تم حفظها به عند الحاجة إلى ذلك . والملف الذي ينشأ نتيجة هذا الأمر يأخذ اسما ممتدا هو "mem." .

في حالة اشتمال الأمر على الاختيار<skeleton> ALL LIKE فإن محتويات الذاكرة التي تتطابق مع الحروف المكتوبة بالأمر بعد كلمة LIKE فقط هي التي تخزن على الملف . أما إذا اشتمل الأمر على الاختيار<skeleton> ALL EXCEPT فإن جميع محتويات الذاكرة ما عدا التي تتطابق مع الحروف المكتوبة بعد كلمة EXCEPT تخزن على الملف. لا يمكن حفظ المصفوفات على ملف MEM .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــال :

(١) لكي تحفظ جميع محتويات الـذاكـرة Memory Variables التي تبـدأ بالحرف S على ملف خارجي باسم STMEM.mem على نفس الوحدة المخصصة معك أدخل الأمر التالي :

```
. SAVE ALL LIKE S* TO STMEM
```

(٢) لكي تحفظ جميع محتويات الذاكرة ما عدا تلك التي يكون الحرف الثالث فيها S على ملف اسمه STMEM1.mem . ويفرض أنك تريد أن تحفظ الملف على وحدة غير المخصصة معك ولتكن A استخدم الأمر:

```
SAVE ALL EXCEPT ??S* TO A:STMEM1
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

RESTORE – STORE

الأمر SAVE SCREEN

يحفظ شاشة موجودة بالذاكرة أو في حقل ذاكرة.

الشكل العام:

SAVE SCREEN [TO <*memvar*>]

حيث:

TO <*memvar*> : حقل الذاكرة الذي ستحفظ به الشاشة.

الشرح:

يستخدم هذا الأمر لحفظ الشاشة الموجودة بألوانها إما داخل الذاكرة أو حقل ذاكرة ليمكنك استرجاعها فيما بعد باستخدام أمر RESTORE SCREEN .

الاختلاف عن *dBASE III PLUS* : غير موجود بها.

مثال:

المثال التالي يحفظ الشاشة الموجودة داخل حقل ذاكرة اسمه SCRVAR

SAVE SCREEN TO SCRVAR

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة:

RESTORE SCREEN – SAVE

الأمر SEEK

يضع المؤشر عند أول سجل يتطابق مع العبارة المذكورة في الأمر داخل ملف مفهرس.

الشكل العام:

SEEK *<expression>*

حيث:

<expression> : عبارة حرفية أو رقمية أو تاريخية.

الشرح:

يبحث هذا الأمر داخل ملف مفهرس عن أول سجل يشتمل على العبارة المذكورة بالأمر. وهو أمر مشابه لأمر FIND إلا أنه أكثر كفاءة فيمكنك أن تبحث عن أي عبارة سواء كانت حرفية أو رقمية أو تاريخية ولابد من وجود علامة التنصيص "" إذا اشتمل الأمر على عبارة حرفية.

إذا وجدت قاعدة البيانات السجل المطلوب فإن المؤشر ينتقل إلى هذا السجل داخل الملف وتصبح الوظيفة FOUND( ) صحيحة (.T.) . أما إذا لم تجد السجل المطلوب وكان أمر SET SOFTSEEK في حالة OFF فإن المؤشر يصل إلى نهاية الملف وتصبح الوظيفة EOF( ) صحيحة (.T.) والوظيفة FOUND( ) خاطئة (.F.) .

وأمر SET SOFTSEEK أمر جديد في «كلبر». وهذا الأمر له حالتان الأولى OFF وهي الوضع التلقائي للأمر، والثانية ON وهي الوضع الذي يمكن الانتقال إليه. فإذا كان في حالة ON ولم تجد «كلبر» السجل الذي تبحث عنه داخل الملف فإن السجل يوضع عند أقرب سجل يلي السجل الذي تبحث عنه وليس في نهاية الملف. وبالتالي فإن كلا من الوظيفة FOUND( ) والوظيفة EOF( ) ستكون خاطئة (.F.) برغم أن السجل غير موجود بالملف.

وعادة يتم البحث داخل الملف المفهرس عن العبارة المتطابقة مع العبارة الموجودة في الأمر. فإذا وجدتها قاعدة البيانات اعتبرت أن العبارة موجودة وأوقفت المؤشر عند هذا السجل فمثلا إذا كنت تبحث عن كلمة ATT في حقل COMPANY وأدخلت الأمر بهذه الصورة:

SEEK A

فستضع المؤشر عند أول سجل يشتمل على الكلمة التي تبدأ بحرف A فإذا وجدت AST مثلا اعتبرت أن الشرط قد تحقق وأوقفت البحث في حين أننا نبحث عن ATT . فإذا كنت تريد التطابق التام أثناء البحث أي إذا كنت تريد أن يستمر البحث حتى تجد كلمة ATT كلها فيجب أن تضع أمر EXACT في وضع ON هكذا:

SET EXACT ON

الاختلاف عن *dBASE III PLUS* : لا يوجد في «دي بيس» أمر SET SOFTSEEK ولذلك فإن المؤشر يوضع دائما في نهاية الملف إذا لم تجد السجل المطلوب.

مثـــال:

المثال التالي يستخدم ملف STOCK.dbf حيث يبحث أول أمر SEEK عن سعر غير موجود بالملف. وتلاحظ أن المؤشر موجود عند أقرب سعر نبحث عنه وليس في نهاية الملف.

```
USE STOCK INDEX IPRICE      && افتح ملف STOCK.DBF مرتبا طبقا ل IPRICE
SEEK 140.500                && في حال عدم وجود السعر 140.500
? FOUND() , EOF()           && ستكون الإجابة: .F.   .T.
SET SOFSEEK ON
SEEK 140.500                && في حالة عدم وجود السعر 140.500
? FOUND() , EOF()           && ستكون الإجابة: .F.   .F.
? PRICE                     && ستحصل على أقرب سعر وهو 144.500
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

FIND – INDEX – SET EXACT – SET DELETED – SET INDEX

SET SOFTSEEK – SOUND( ) – EOF( )

الأمـر SELECT

يختار إحدى المناطق العاملة (Work Areas) ليضع فيها ملف قاعدة البيانات.

الشكل العام:

SELECT <*work area*>/<alias>/(<*expN*>)

حيـــث:

<work area> : إما رقم من صفر إلى ٢٥٤ .

<alias> : اسم ملف قاعدة البيانات أو اسم بديل له يعرف للحاسب في أمر USE بعد كلمة alias .

<expN> : قيمة رقمية تقع بين صفر و٢٥٤ .

الشـــرح:

يستخدم هذا الأمر لتختار ملف قاعدة البيانات وتضعه في إحدى المناطق العاملة بالذاكرة. وجدير بالذكر أن الذاكرة يمكن تقسيمها حتى ٢٥٤ منطقة ويمكنك اختيار منطقة (Workarea) إما بتحديد رقمها أو الحرف الدال عليها (من A إلى J) أو باختيار الاسم البديل <alias> . فإذا اخترت المنطقة رقم صفر (SELEC 0) فتخصص لك «كلبر» أول منطقة غير مستخدمة. ويمكن أن يوضع في كل منطقة ملف قاعدة بيانات و١٥ فهرسا متصلة به.

ورغم أنـك تستـطيع تقسيم الذاكرة حتى ٢٥٤ منطقة ويمكنك وضع ملف قاعدة بيانات في كل منطقة بيد أنك لا تستطيع أن تتعامل في نفس اللحظة إلا مع ملف واحد فقط والباقي يوضع في المنطقة الخلفية (Background Workarea) . ولذلك فإن أوامر ووظائف قاعدة البيانات التي تستخدم لتحريك المؤشر داخل الملف مثل الأمر SKIP أو الوظيفة DBF( ) تؤثر فقط على الملف المختار في المنطقة المختارة.

الاختلاف عن *dBASE III PLUS* : تتعامل «دي بيس» مع عشرة مناطق فقط.

مثـــال:

هذا المثال يخصص ملف STOCK.dbf للمنطقة A وملف SALE.dbf للمنطقة B وملف PURCH.dbf للمنطقة C :

```
CLEAR ALL                اغلق جميع الملفات وارجع الى المنطقة رقم 1 &&
USE STOCK INDEX ITEM_NO
SELECT B
USE SALE INDEX INV_NO,ITEM_NO,IDATE
SELECT C
USE PURCH INDEX ITEM_NO
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET INDEX – USE – ALIAS( )

الأمـــر SET ALTERNATE

يسمـح SET ALTERNATE on/OFF أو لا يسمـح بإرسال المخرجات التي تظهر على الشاشة إلى ملف نصي (.TXT) بينما يعرِّف أمر SET ALTERNATE TO اسم الملف الذي ستوضع عليه هذه المخرجات.

الشكل العام:

SET ALTERNATE on/OFF /(<*expL*>)

SET ALTERNATE TO [<*filename*>/(<*expC*>)]

حـــيث:

<*filename*>/(<*expC*>) : اسم الملف النصي.

<*expL*> : قيمة منطقية .T. أو .F. للدلالة على حالة الأمر.

الشــــرح:

الوضع الأصلي لهذه الدالة هو لا (OFF) . إذا كانت هذه الدالة في وضع نعم (ON) فإن المخرجات التي تظهر على الشاشة ترسل إلى الملف النصي المذكور اسمه في أمـر SET ALTERNATE TO فإذا أردت أن توقف تسجيل المخـرجات على الملف النصي فيكفي أن تعيد الدالة إلى وضع لا بالأمر:

SET ALTERNATE OFF

ويأخذ الملف النصي اسما ممتدا هو ".txt" .

إذا أردت أن تغلق هذا الملف فيكفي أن تصدر الأمر بدون اسم الملف هكذا:

SET ALTERNATE TO

الاختلاف عن *dBASE III PLUS* : «دي بيس» لا تستخدم الاختيار <*expL*>

مثــــال:

يوضح المثال التالي كيف تنشىء ملفا نصيا باسم SCRTXT.txt لتضع عليه مخرجات الشاشة وكيف تفتحه وتغلقه.

```
SET ALTERNATE TO SCRTXT          && فتح ملفا نصيا لقبول مخرجات الشاشة
SET ALTERNATE ON                 && ابدأ تسجيل مخرجات الشاشة
SET ALTERNATE OFF                && اوقف تسجيل مخرجات الشاشة
SLOSE ALTERNATE                  && اغلق الملف النصي المفتوح
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

CLOSE

الأمر SET BELL

يسمح أو لا يسمح بسماع صوت الجرس أثناء إدخال البيانات.

الشكل العام:

SET BELL on/OFF/(<*expL*>)

حيث:

<*expL*> : تعبير منطقي (.T.) أو (.F.) بدلاً من ON أو OFF .

الشرح:

الوضع الأصلي لهذه الدالة هو لا (OFF) . ويعني وضع هذه الدالة في وضع ON انذار المستخدم بإصدار صوت الجرس إذا أدخل بيانات غير صحيحة أو إذا وصل إلى نهاية الحقل أثناء إدخال البيانات. فإذا أردت إلغاء هذا الصوت فيجب أن تضع الدالة في وضع لا بالأمر:

SET BELL OFF

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» تغيير منطقي (<*expL*>)

مثال:

في المثال التالي سيسمع المستخدم صوت الجرس إذا أدخل رقماً لتنبيهه عن وجود خطأ بينما إذا أدخل حرفاً ردا على الرسالة فلن يسمع هذا الصوت.

```
OPTION = " "
SET BELL ON

@ 24,2 SAY "O.K. to continue...." GET OPTION
READ
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET CONFIRM – CHR( ) – TONE( )

الأمر SET CENTURY

يسمح أو لا يسمح بإظهار القرن في حالة إظهار التاريخ .

الشكل العام:

SET CENTURY on/OFF/(<*expL*>)

حيـــث:

(<*expL*>) : عبارة منطقية .T. أو .F. بدلاً من ON أو OFF .

الشـــرح:

الـوضـع الأصـلي لهذا الـدالة هو لا OFF . ويعني هذا الوضع إظهار السنة بالرقمين الأوليين من جهة اليمين الدالين عليها بدون إظهار الرقمين الممثلين للقرن فمثلا سنة 90 تعني 1990 . ويمكنك تغيير هذا الوضع إلى وضع نعم (ON) وتعني إظهار القرن ضمن حقول التاريخ سواء في حالة إدخال بيانات إلى حقل تاريخي أو استرجاع بيانات من حقل تاريخي . فإذا لم يدخل القرن إلى حقل التاريخ منذ البداية فإن قاعدة البيانات تعتبره القرن العشرون عند إجراء عمليات حسابية خاصة بهذا الحقل مثل طرح تاريخ من تاريخ آخر أما إذا أدخلت البيانات منذ البداية بقرن آخر مثل ١٤١١ وهو العام الهجري المقابل للعام الميلادي ١٩٩٠ فإن قاعدة البيانات تجري العمليات الحسابية بناء على القرن المسجل ضمن حقل التاريخ .

وتجدر الاشارة إلى أنه رغم أن التاريخ يظهر مكونا من عشرة حروف إلا أنه يظل يشغل مساحة قدرها ٨ حروف في الملف الأصلي.

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» التعبير المنطقي (<*expL*>) .

مثـــال :

يظهر المثال التالي الحقل التاريخي بصورتين: الأولى بدون القرن والثانية تظهر القرن ضمن حقل التاريخ .

```
MYDATE=DATE()
? MYDATE                && 10/23/90   :الاجابة
SET CENTURY ON
? MYDATE                && 10/23/1990   :الاجابة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

CTOD( ) – DTOC( ) – DATE( ) – YEAR( ) – SET DATE

الأمر SET COLOR

الأمر SET COLOR ON/off يتعامل مع المخرجات على أن الشاشة ملونة أو غير ملونة والأمر SET COLOR TO يغير ألوان الشاشة حسب طلبك.

الشكل العام:

SET COLOR ON/off

SET COLOR TO [<*standard*> [,<*enhanced*>] [,<*border*>] [,<*Unselected*>]]/(<*expC*>)

حيث:

<*standard*>	:	ألوان الشاشة الأمامية.
<*enhanced*>	:	ألوان الشاشة العكسية (Reverse Video)
<*border*>	:	جوانب الشاشة.
<*Unselected*>	:	تظهر حقل GET الموجود تحت المؤشر بألوان الشاشة العكسية.
<*expC*>	:	عبارة بديل لاختيارات الألوان.

الشـــرح:

أمر SET COLOR TO فيسمح بتغيير ألوان الشاشة حسب رغبتك إذا لم ترق لك الألوان التي تم ضبطها بوساطة قاعدة البيانات. وإليك تفصيل ذلك:

: <*standard*>

تمثل ألـوان الشاشة الأمامية والخلفية ويستخدم حرف أو حرفين للدلالة على اللون المطلوب ـ كما ستعرف بعد قليل ـ ولذلك يعوض عنها في الأمر بحرف أو حرفين بينها هذه العلامة «/» ويمثل الجزء الأيسر من العلامة أمامية الشاشة (Foreground) أي لون الكتابة ويمثل الجزء الأيمن من العلامة خلفية الشاشة (Background) أي لون الشاشة.

: <enhanced>

تمثل ألوان الشاشة العكسية (Reverse Video) الأمامية والخلفية وأيضا يعوض عنها في الأمر بحرف أو حرفين بينها العلامة / .

ويمثل الجزء الأيسر من العلامة أمامية الشاشة ويمثل الجزء الأيمن من العلامة خلفية الشاشة وأقصد بالشاشة العكسية (Reverse Video) الشاشة التي تظهر مع أوامر GET...@ . وهي المنطقة المسموح فيها بالكتابة.

: <border>

وهي الجوانب ويعوض عنها في الأمر بحرف أو حرفين للدلالة على اللون المطلوب وإليك بيان الألوان المسموحة في قاعدة البيانات والحروف الدالة عليها.

الحرف	اللــــون
N	أسود (Black)
B	أزرق (Blue)
G	أخضر (Green)
BG	سماوي (Cyan)
R	أحمر (Red)
RB	أحمر فاتح (Magenta)
GR	بني (Brown)
W	أبيض (White)
N+	رمادي (Gray)
GR+	أصفر (Yellow)
X	إخفاء الكتابة (Blank)

ملاحظـــات :

١ ـ إذا استخدم الأمر بصيغة SET COLOR TO فقط فستظهر لك الألوان التي تم ضبطها مسبقا بوساطة قاعدة البيانات .

٢ ـ إذا استخـدمت علامـة * بعـد الحرف فإن الكتابة تومض (تظهر وتختفي) على الشاشة وإضافة علامة + بعد الحرف تجعل اللون يظهر حادا.

الاختلاف عن dBASE III PLUS : يستخدم «كلر» الكلمة الانجليزية COLOUR أيضا واختياران هما Unselected (<*expC*>)

مثـــال :

للكتابة بلون أبيض على شاشة زرقاء وجعل لون الشاشة العكسية أسود والكتابة عليها بالأصفر استخدم المثال التالي :

```
STORE "W+/B,GR+/N" TO MYCOLOR
SET COLOR TO (MYCOLOR)
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

SET – SET INTENSITY – ISCOLOR( )

الأمر SET CONFIRM

يتطلب أو لا يتطلب الضغط على مفتاح الادخال Enter في حالة إدخال البيانات.

الشكل العام:

SET CONFIRM on/OFF/(<*expL*>)

حيث:

<*expL*> : عبارة منطقية : .T. أو .F. بدلا من ON أو OFF .

الشرح:

الوضع الأصلي لهذه الدالة هو OFF وهذا الوضع يسمح بانتقال مؤشر الشاشة إلى الحقل التالي مباشرة بدون الضغط على مفتاح الادخال Enter عندما تمتلىء المساحة المتاحة لادخال بيانات هذا الحقل وذلك مع الأوامر التي تستخدم لادخال بيانات إلى الحاسب مثل GET... @ .

أما الوضع البديل وهو SET CONFIRM ON فيتطلب منك ضغط مفتاح الادخال Enter ليتم إدخال بيانات إلى الحاسب سواء امتلأت المساحة المتاحة لادخال بيانات الحقل أو لم تمتلىء.

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» التعبير الحرفي (<*expC*>) .

مثال:

نتيجة لتنفيذ الأوامر التالية فإن المستخدم يجب أن يضغط مفتاح الادخال Enter بعد كتابة الحرف Y أو الحرف N ليدخل إلى الحاسب.

```
SET CONFIRM ON
STORE SPACE(1) TO YN
@ 24,2 SAY "O.K. to continue? [Y/N]" GET YN
READ
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@...SAY...GET – SET BELL – READ

الأمر SET CONSOLE

يسمح بإظهار المخرجات على الشاشة أو يلغي هذه الامكانية.

الشكل العام:

SET CONSOLE ON/off/(<*expL*>)

حيث:

<*expL*> : عبارة منطقية .T. أو .F. بدلاً من ON أو OFF .

الشرح:

الوضع الأصلي لهذه الدالة هو نعم (ON) ويعني إرسال جميع مخرجات البرنامج إلى الشاشة ولذلك فإن هذا الأمر يستخدم فقط داخل البرنامج. ويجب الانتباه إلى أن المخرجات نتيجة أمر SAY...GET...@ تظهر على الوحدة المختارة بالأمر SET DEVICE ولا تأثير لأمر SET CONSOLE على هذه المخرجات. فإذا كنت تريد إظهار تقرير مثلا على الطابعة بدون أن يظهر على الشاشة فيجب أن تصدر مع أمر SET CONSOLE OFF أمرا آخر هو:

SET DEVICE TO PRINT

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» التعبير (<*expL*>) .

مثال:

يسمح المثال التالي بإظهار التقرير STKRPRT.frm على الطابعة فقط في حين تظهر رسالة "Report Printing.....Please Wait" على الشاشة.

```
USE STOCK
SET CONSOLE ON      && اسمح للشاشة باظهار المخرجات
@ 24,2 SAY "Report printing... Please wait "
SET CONSOLE OFF     && اوقف اظهار المخرجات على الشاشة اثناء طباعة التقرير
REPORT FORM STKRPRT TO PRINT
SET CONSOLE ON      && اعد اظهار المخرجات على الشاشة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

@...SAY – SET DEVICE

الأمر SET CURSOR

يظهر أو يمنع إظهار مؤشر الشاشة.

الشكل العام:

SET CURSOR ON/off/(<*expL*>)

حيث:

<*expL*> : عبارة منطقية .T. أو .F. بدلاً من ON أو OFF .

الشرح:

يظهر مؤشر الشاشة تلقائيا فإذا أردت إخفاءه ضع الأمر في حالة OFF ويفيد ذلك في بعض الحالات منها قبل استخدام أمر MENU TO لاخفاء المؤشر أثناء تحريك الشريط المضاء بين اختيارات القائمة. يجوز أن تستخدم تعبيرا منطقيا (.T.) أو (.F.) . للدلالة على الحالة التي ترغبها.

الاختلاف عن *dBASE III PLUS*: لا تستخدم «دي بيس» التعبير (<*expL*>) .

مثال:

المثال التالي سيسمح بإدخال رقم في حقل MVAL بدون إظهار المؤشر.

```
MVAL=0
SET CURSOR OFF
@ 12,10 SAY "Enter your number: " GET MVAL
READ
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET CONSOLE

الأمر SET DATE

يحدد الشكل الذي سيظهر به التاريخ .

الشكل العام:

SET DATE AMERICAN/ANSI/BRITISH/ITLIAN/FRENCH/GERMAN

الشـــرح:

الوضع الأصلي للشكل الذي يظهر به التاريخ هو AMERICAN أو الاستخدام الأمريكي بالصورة التالية : mm/dd/yy

أي رقمان من الشمال للدلالة على الشهر يليهما علامة / يليها رقمان للدلالة على اليوم تليهما علامة / يليها رقمان للدلالة على السنة . بيد أنك تستطيع تغيير هذا الشكل . وإليك بيان الاختيارات المتاحة والشكل الذي يظهر مع كل منها .

الاختيار	شكل التاريخ
AMERICAN	mm/dd/yy
ANSI	yy.mm.dd
BRITISH	dd/mm/yy
ITALIAN	dd–mm–yy
FRENCH	dd/mm/yy
GERMAN	dd.mm.yy

ولعل أقربها للاستخدام العربي للتاريخ هو ANSI فهو يظهر من اليمين إلى الشمال اليوم ثم الشهر ثم السنة .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــال:

توضح الأمثلة التالية بعض الأشكال المستخدمة لاظهار التاريخ .

```
MYDATE = DATE()
? MYDATE                      && 10/23/90   الاجابة:
SET DATE ANSI
? MYDATE                      && 90.10.23   الاجابة:
SET DATE BRITISH
? MYDATE                      && 23/10/90   الاجابة:
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

DATE( ) - SET CENTURY

الأمـر SET DECIMALS

يحدد عدد الأرقام العشرية التي تظهر بعد العلامة العشرية في العمليات الحسابية.

الشكل العام:

SET DECIMALS TO <expN>

حيـث:

<expN> : أي رقم أو تعبير رقمي مفهوم لقاعدة البيانات.

الشـرح:

يستخدم هذا الأمر لتحديد الحد الأدنى لعدد الأرقام العشرية التي تظهر بعد العلامة العشرية في العمليات الحسابية الناتجة عن عمليات القسمة أو الناتجة من الوظائف التالية:

EXP( ) – LOG( ) – SQRT( ) – VAL( )

والعدد التلقائي للأرقام التي تظهر بعد العلامة العشرية ما لم تستخدم هذا الأمر هو ٢.

لكي تحصل على عدد من الأرقام العشرية في جميع العمليات الحسابية ـ وليست عمليات القسمة فقط ـ مساويا للعدد المختار في أمر SET DECIMALS يجب أن تستخدم أمر SET FIXED ON.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـال:

يظهر المثال التالي الأرقام العشرية بالعدد المحدد سلفا بواسطة قاعدة البيانات.

```
? 34/564                    && 0.06   الإجابة:
? 34.000/564.000            && 0.060  الإجابة:
? LOG(3)                    && 1.10   الإجابة:
```

في حين يستخدم المثال التالي الأمر **SET DECIMALS**

```
SET DECIMALS TO 5
? 34/564                    && 0.06028   الإجابة:
? 34.000/564.000            && 0.06028   الإجابة:
? LOG(3)                    && 1.09861   الإجابة:
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET FIXED – EXP( ) – LOG( ) – SQRT( ) – VAL( )

الأمر SET DEFAULT

يحدد مشغل القرص الذي ستستخدمه قاعدة البيانات للبحث عن أو لتخزين ملفاتها.

الشكل العام:

SET DEFAULT TO <*drive*> [:<*path*>]

حيث:

<*drive*>[:<*PATH*>] : حرف يمثل مشغل القرص والطريق الذي تسلكه «كلبر» للبحث عن الملفات.

الشرح:

يستخدم هذا الأمر لتحديد اسم مشغل القرص والدليل الذي تستخدمه قاعدة البيانات للبحث عن أو لتخزين ملف إذا لم يشتمل الأمر على اسم مشغل القرص والدليل. وذلك في الأوامر التي تتطلب البحث عن ملف مثل:
COPY – USE – DIR – ERASE – RENAME . . . الخ.

أما إذا اشتمل الأمر على اسم مشغل القرص فإن قاعدة البيانات تبحث في القرص المحدد.

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» PATH .

مثال:

لكي تفتح ملف STOCK.dbf الموجود تحت الدليل \CLIPPER\APP على الوحدة C استخدم الأوامر التالية:

```
SET DEFAULT TO C:\CLIPPER\APP
USE STOCK
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET PATH

الأمر SET DELETED

يسمح بإظهار السجلات المعلمة لأغراض الحذف أو قد لا يسمح بذلك.

الشكل العام:

SET DELETED on/OFF/(<expL>)

حيث:

<expL> : عبارة منطقية .T. أو .F. بدلاً من ON وOFF .

الشرح:

الوضع الأصلي لهذه الدالة هو لا OFF وتعني إظهار السجلات المعلمة بنية الحذف مسبوقة بعلامة * (مع بعض الأوامر).

أما ON SET DELETED فإنها تمنع إظهار السجلات المعلمة بنية الحذف مع مثل هذه الأوامر. أما تلك الأوامر التي تنفذ على السجل الذي يقف عنده المؤشر مثل أمر <n> DISPLAY RECORD أو <n> GO/GOTO فإنها تتعامل مع السجلات المعلمة بنية الحذف بدون أن تتأثر بهذا الأمر.

إذا كانت هذه الدالة في وضع نعم (ON) فإن أمر RECALL ALL لن يسترجع أية سجلات.

لاحظ أن أمر INDEX وأمر REINDEX يتعامل مع جميع سجلات الملف سواء كانت الدالة في وضع ON أو OFF .

الاختلاف عن *dBASE III PLUS* : لا تتعامل «دي بيس» مع التعبير المنطقي (<expL>) .

مثـــال:

لكي تستبعد السجلات المعلمة لأغراض الحذف عند تجميع بيانات السجلات بالأمر SUM استخدم المثال التالي:

```
USE STOCK
DELETE RECORD 3
SUM PRICE                 && ستشمل عملية الجمع كل السجلات &&
SET DELETED ON
SUM PRICE                 && سيستبعد سجل من عملية الجمع &&
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

DELETED( )

الأمر SET DELIMITERS

يستخدم بصيغة SET DELIMTERS on/OFF إما لاظهار فاصل بين الحقول أو لمنع إظهاره.

أما أمر SET DELIMITERS TO فيحدد نوع الفاصل الذي سيستخدم عند إظهار البيانات.

الشكل العام:

SET DELIMITERS on/OFF/(<*expL*>)

SET DELIMITERS TO [<*expC*>/DEFAULT]

حيث:

<*expL*> : قيمة منطقية .T. أو .F. للدلالة على حالة الأمر.

<*expC*> : حرف أو حرفين.

الشرح:

الوضع الأصلي لهذه الدالة هو لا (OFF) وتعني إظهار الحقول التي تستخدم في حالة إدخال البيانات سواء إلى الملف أو إلى الذاكرة بشاشة عكسية بدون علامة التنصيص (:).

أما SET DELIMITERS ON فإنها تظهر هذه الحقول بألوان عكسية (Reverse Video) أيضا ولكنها تضع علامتي التنصيص (:) على حافتي الحقل (أي قبل أول وبعد آخر حرف متاح لادخال البيانات).

ويستخدم الأمر SET DELIMITERS TO <*expC*> لاستبدال علامتي التنصيص (:) التي تظهر عند بداية ونهاية الحقل بالحرف أو الحرفين الموجودين بالأمر.

ويمكن أن تكون <expC> حرف واحد أو حرفين فإذا استبدلت بحرف واحد فإن هذا الحرف سيظهر عند بداية ونهاية الحقل. أما إذا إستبدلت بحرفين فإن الحرف الأول سيظهر عند بداية الحقل والحرف الثاني عند نهايته.

إذا أردت أن تستخدم علامتي التنصيص المحددة بواسطة قاعدة البيانات بدلا من الحرف أو الحرفين المستخدمين مع أمر SET DELIMITERS فيجب أن تستخدم الأمر بصورة SET DELIMITERS TO DEFAULT .

الاختلاف عن dBASE III PLUS : لا تستخدم «دي بيس» التعبير (<expL>)

مثـــال :

المثال الآتي يوضح تأثير أمر SET DELIMITERS فهو يظهر هذه الأقواس [ ] لتحديد بداية ونهاية البيانات التي ستدخل إلى الذاكرة باستخدام أمر GET...@ .

```
SET DELIMITERS ON
SET DELIMITERS TO "[]"
STORE SPACE (14) TO M_NAME
@ 5,5 GET M_NAME
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

@... – APPEND – CHANGE – EDIT – INSERT – READ – SET INTESITY

الأمر SET DEVICE

يسمح بإظهار مخرجات أمر @...SAY إما على الطابعة أو على الشاشة.

الشكل العام:

SET DEVICE TO SCREEN/print

الشـــرح:

الـوضـع الأصلي لهذه الدالة هو SCREEN ويعني أن نتائج الأمر @...SAY ستـظهـر على الشـاشـة في العمـود والسـطر المحـددين أمـا الـوضـع البـديل وهـو SET DEVICE TO PRINT فيعني أن نتائج الأمر ستظهر على الطابعة بدلا من الشاشة وفي هذه الحالة فإذا اشتمل الأمر @ على الاختيار GET فإن قاعدة البيانات ستهمل هذا الاختيار حيث لا يصلح إدخال بيانات بواسطة الطابعة.

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثـــال:

في المثال التالي ستظهر عبارة "I can see you" على الشاشة أما عبارة "It is there" فستظهر على الطابعة.

```
@ 1, 1 SAY "I can see you"
SET DEVICE TO PRINT        &&  توجيه مخرجات أمر @ الى الطابعة
@ 1, 1 SAY "It is there"
SET DEVICE TO SCREEN       &&  توجيه مخرجات أمر @ الى الشاشة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@ – EJECT – SET FORMAT – PCOL( ) – PROW( )

يسمح بإلغاء البرنامج أثناء تنفيذه حال الضغط على مفتاح Esc أو قد يلغي هذه الامكانية.

الشكل العام:

SET ESCAPE ON/off/(<*expL*>)

حيـــث:

<*expL*> : عبارة منطقية .T. أو .F. بدلا من ON أو OFF.

الشـــرح:

الوضع الأصلي لهذه الدالة هو نعم (ON) وهي تمكنك من إلغاء تنفيذ أمر READ بمجرد ضغط مفتاح Esc.

أيضا ضغط مفتاح Alt-C أثناء تنفيذ البرنامج يلغي التنفيذ.

إذا أردت أن توقف تأثير مفتاح Esc على الأوامر والبرنامج ضع هذه الدالة في وضع لا (OFF).

الاختلاف عن *dBASE III PLUS*: لا تستخدم «دي بيس» مفتاح Alt-C لالغاء تنفيذ البرنامج وتستخدم بدلا منه مفتاح Esc.

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

INKEY( ) – REA – SET KEY

SET EXACT الأمر

يطلب من قاعدة البيانات مطابقة السؤال على بيانات الحقل الذي يتم البحث فيه مطابقة تامة .

الشكل العام:

SET EXACT on/OFF/(<*expL*>

حيث :

(<*expL*>) : تعبير منطقي .T. أو .F. بدلاً من ON أو OFF .

الشرح :

الوضع الأصلي لهذه الدالة هو لا (OFF) ويعني أن قاعدة البيانات ستعتبر المطابقة صحيحة في حالة تطابق الحروف الموجودة على يمين علامة المقارنة (علامة = مثلا) مع الحروف الأولى من العبارة الموجودة على يسار هذه العلامة بصرف النظر عن باقي الحروف .

أما الوضع البديل وهو SET EXACT ON فيتطلب من قاعدة البيانات أن تعتبر المقارنة صحيحة إذا تطابقت جميع الحروف الموجودة في العبارتين .

وتتأثر جميع الأوامر التي تستخدم للاستفسار في قاعدة البيانات مثل :
LOCATE – SEEK – FIND – FOR – WHILE بالحالة التي عليها هذه الدالة .

الاختلاف عن dBASE III PLUS : لا تستخدم «دي بيس» التعبير (<*expL*>) .

مثال :

يظهر المثال التالي جميع الطلاب الذين تبدأ أسماؤهم بالحروف "ABDU" (مثلا ABDU, ABDULLAH, ABDULAZIZ) .

```
SET EXACT OFF
LIST FOR FIRSTNAME = "ABDU"
```

في حين يظهر المثال التالي الطالب أو الطلاب الذين تتكون أسماؤهم من ٤ حروف هي "Abdu" فقط.

```
SET EXACT ON
LIST FOR FIRSTNAME = "ABDU"
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

LOCATE - FIND - SEEK - =

الأمر SET EXCLUSIVE

يسمح لأكثر من مستفيد في حالة استخدام شبكة اتصالات محلية باستخدام نفس الملف أو يمنع ذلك.

الشكل العام:

SET EXCLUSIVE ON/off/(<*expL*>

حيث:

(<*expL*>) : تعبير منطقي .T. أو .F. بدلاً من ON أو OFF .

الشرح:

الوضع الأصلي لهذا الأمر نعم (ON) وتعني أن أكثر من مستفيد لا يسمح لهم بفتح نفس الملف والوضع البديل هو لا (OFF) ويعني السماح لأكثر من مستفيد في شبكة الاتصالات بفتح نفس الملف والتعامل معه. ولذلك فإذا جعلت هذا الأمر في وضع OFF فيجب أن تتبع قواعد التعامل مع شبكة الاتصالات عند تعديل البيانات والاطلاع عليها. (راجع الفصل الحادي عشر استخدام شبكات الاتصالات) لتجنب المشاكل التي تنتج من تعديل بيانات الملف بواسطة أكثر من مستفيد في الشبكة.

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» التعبير (<*expL*>) .

مثال:

```
SET EXCLUSIVE OFF              && اجعل الملفات التالية مشاعة &&
USE EMPLOYEE
IF NETERR()                    && اذا حدث خطأ اثناء فتح الملف &&
   ? "Network error... File not available"
   CLEAR
   CLEAR ALL
   RETURN
ENDIF
SET INDEX TO EMP_ID
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

USE – EXCLUSIVE – FLOCK( ) – NETERR( ) – RLOCK( )

الأمـــر SET FILTER

يجعل الملف يبدو كما لو كان مشتملا فقط على السجلات المتطابقة مع شروط محددة.

الشكل العام:

SET FILTER TO <*condition*>

حيـــث:

<*condition*> : الشرط الذي يحدد نوعية السجلات التي ستختار.

الشـــرح:

يستخـدم هذا الأمر لاختيار مجموعة سجلات من بين سجلات ملف قاعدة البيانات لتكون هي المتاحة عند التعامل مع الملف (سواء باسترجاع بيانات الملف أو البحث داخـل الملف. . . الخ). وبإمكانك في حالة فتح أكثر من ملف أن تنشىء مصفاة (Filter) لكل ملف على حده. ولذلك يجب أن تنتبه إلى أن الأمر سينفذ مع الملف المختار في المنطقة المختارة، فمثلا:

```
SELECT B
SET FILTER TO CITY = "RIYADH"
```

تؤثر فقط على الملف الموجود بالمنطقة (Workarea) B داخل الذاكرة.

والأمر SET FILTER TO <*condition*> يجعل ملف قاعدة البيانات يبدو كما لو كان مشتملا على السجلات المتطابقة مع الشرط فقط ولذلك فإن جميع الأوامر التي تتعامـل مع الملف مثل (SUM – LIST – AVERAGE – REPORT) تتعامل معه بصورته الجديدة.

يجب أن تحرك المـؤشر داخـــل ملف قاعـــدة البيانـــات بعـــد إصـــدار أمـــر SET FILTER وذلك بإصدار أمر GO TOP بعده مباشرة ليتأثر الملف بالشروط الجديدة

إلا إذا كنت تستخدم أحد الأوامر التي تضع المؤشر عند بداية الملف تلقائيا مثل أمر LIST .

الاختلاف عن *dBASE III PLUS* : لا تستخدم «كلبر» الاختيار FILE <*query file*> لأنه لا يوجد به أمر CREATE/MODIFY QUERY

مثال :

يستخدم المثال التالي ملف STOCK.dbf ويتعامل معه كما لو كان يشتمل على سجلات شركة IBM فقط .

```
USE STOCK
SET FILTER TO COMPANY = "IBM"
GO TOP
LIST ACCOUNTNO,COMPANY          && ستظهر سجلات IBM فقط
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

SET DELETED

الأمر SET FIXED

يحدد عدد الأرقام العشرية التي تظهر في العمليات الحسابية.

الشكل العام:

SET FIXED on/OFF/(<*expL*>)

حيث:

(<*expL*>) : تعبير منطقي .T. أو .F. بدلاً من ON أو OFF .

الشرح:

يستخدم هذا الأمر لتحديد عدد الأرقام العشرية التي ستظهر بعد العلامة العشرية في العمليات الحسابية. الوضع الأصلي لهذه الدالة هو لا (OFF) وفيه تبدو الأرقام بعد العلامة العشرية طبقا لقواعد الحساب المعروفة. أما في الوضع البديل SET FIXED ON فإن العمليات الحسابية تظهر عددا من الأرقام العشرية بعد العلامة العشرية مساويا للعدد المحدد في أمر SET DECIMALS ، أو العدد التلقائي - إذا لم يستخدم أمر SET DECIMALS - وهو ٢ .

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» التعبير (<*expL*>)

مثال:

يظهر المثال التالي نتيجة إجراء عمليات حسابية مرة والأمر في حالة لا (OFF) ومرة أخرى والأمر في حالة نعم (ON) .

```
? 45.76767+40.65667          && 86.42434   الإجابة:
SET FIXED ON
? 45.76767+40.65667          && 86.42    الإجابة:
SET DECIMAL TO 3
? 45.76767+40.65667          && 86.424   الإجابة:
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET DECIMALS

الأمــر SET FORMAT

يخصص ملف إدخال بيانات سبق إعداده لأول أمر READ يأتي بعده في البرنامج .

الشكل العام:

SET FORMAT TO <*procedure*>

حيـــث :

<*procedure*> : اســم ملف شاشـــة الادخـــال (fmt. أو prg. أو procedure.) .

الشـــرح :

يختلف هذا الأمـر شيئـا ما عنـه في dBASE III PLUS . فلا تمسح Clipper الشاشة تلقائيا عند استدعاء هذا الملف. ولذلك فننصح إذا كنت تنشيء هذا الملف باستخدام أمـر CREATE SCREEN في «دي بيس» أن تغير اســم الملف النصـي (FMT.) الــذي ينشــأ من FMT. إلى PRG. وتضيف إليه الأوامــر الــلازمـة مثـل CLEAR في بدايته.

عيب آخر في استخدام هذا الأمر مع «كلبر» وهو أن «كلبر» تتعامل مع ملف واحد فقط من هذا النوع في نفس اللحظة وليس ملف لكل منطقة مفتوحة .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــال :

المثال التالي عبارة عن برنامج يستدعي إجراء يشتمل على أوامر تجهيز شاشة الادخال .

```
* Program: ENTRY.PRG
USE STOCK
APPEND BLANK
SET FORMAT TO STFORM
READ
*
PROCEDURE STFORM
  CLEAR
  @ 10,05 SAY "Account number  : " GET ACCOUNTNO
  @ 11,05 SAY "Number of shares: " GET NO_SHARES
  @ 12,05 SAY "Price           : " GET PRICE
  @ 13,05 SAY "Date            : " GET DATE
RETURN
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

@ – APPEND – SET DEVICE – SET PROCEDURE

الأمر SET FUNCTION

يخصص قيما لمفاتيح الوظائف.

الشكل العام:

SET FUNCTION <*key*> TO <*expC*> [;]

حيـــث:

<*key*> : رقم المفتاح (من ٢ إلى ٤٠).

<*expC*> : جملة أو عبارة تخزن لحين استدعائها.

الشـــرح:

يمكنـك باستخـدام هذا الأمر تخصيص عبارة (قد تكون أمرا مقبولا بواسطة قاعدة البيانات) لكل مفتاح من مفاتيح الوظائف ما عدا مفتاح F1 فلا يمكن تغييره.

ويوضح الجدول التالي الأرقام من ٢ إلى ٤٠ والمفاتيح المقابلة لها.

رقم المفتاح	المفاتيح المقابلة
٢ ـ ١٠	F2 – F10
١١ ـ ٢٠	Shift - F1 – Shift - F10
٢١ ـ ٣٠	Ctrl - F1 – Ctrl - F10
٣١ ـ ٤٠	Alt - F1 – Alt - F10

الاختلاف عن *dBASE III PLUS*: تسمح «دي بيس» بتخصيص ٩ مفاتيح فقط من F2 إلى F10 .

مثـــال:

لكي تخصص الأمر DO MASTER للمفتاح F9 ليتم تنفيذ الأمر بمجرد ضغط

هذا المفتاح استخدم الأمر التالي:

```
SET FUNCTION 9 TO "DO MASTER" + CHR(13)  &&  تعني مفتاح الادخال CHR(13)
```

ولكي تخصص نفس الأمر لنفس المفتاح بحيث لا يتم تنفيذ الأمر إلا بعد الضغط على مفتاح الادخال Enter أدخل الأمر التالي:

```
SET FUNCTION 9 TO "DO MASTER"
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET KEY

الأمر SET INDEX

يفتح ملفات الفهرسة.

الشكل العام:

SET INDEX TO [<*index file list*>/(<*expC1*>),...]

حيث:

<*index file list*>/(<*expC*>) : أسماء ملفات الفهرسة المطلوب فتحها.

الشرح:

يستخدم هذا الأمر لتخصيص ملف أو أكثر من ملفات الفهرسة مع ملف قاعدة البيانات المفتوح ولا يجب أن تزيد عدد ملفات الفهرسة المفتوحة معا عن ١٥.

إذا استبدلت <*index file list*> بأكثر من اسم ملف فإن الملف المذكور اسمه أولا يعتبر هو الملف الرئيسي الذي يتحكم في ترتيب ملف قاعدة البيانات وهو الذي يوجه أوامر البحث مثل SEEK أو FIND فإذا احتجت لتغيير ملف الفهرسة الرئيسي بآخر مكانه ليكون هو الرئيسي فيجب أن تستخدم أمر: SET ORDER TO

تخصص قاعدة البيانات لملف الفهرس اسما ممتدا هو ".NTX".

صيغة SET INDEX TO بدون اختيارات أخرى تغلق جميع ملفات الفهرسة المفتوحة في المنطقة المختارة.

الاختلاف عن dBASE III PLUS:

- أقصى عدد للملفات الفهرسة التي يمكن فتحها مع ملف قاعدة البيانات في «دي بيس» هو ٧. أما في «كلبر» فهو ١٥.
- لا يتعرف «كلبر» على أمر ? SET INDEX TO

مثال:

المثال الآتي يفتح ملف قاعدة البيانات STOCK.dbf ثم يفتح ملفات الفهرسة التي تخصه بالترتيب الواردة به في الأمر SET INDEX

```
USE STOCK
SET INDEX TO IACC,IDATE,IPRICE
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

CLOSE – INDEX – REINDEX – SET ORDER – USE

الأمـــر SET INTENSITY

يسمح بإظهار ألوان للشاشة العكسية أو يمنع ذلك.

الشكل العام:

SET INTENSITY ON/off/(<*expL*>)

حيـــث:

(<*expL*>) : تعبير منطقي .T. أو .F. بدلا من ON أو OFF .

الشـــرح:

الوضع الأصلي لهذه الدالة هو نعم (ON) . ويعني إظهار ألوان الشاشة بلون مخالف عند إدخال بيانات إلى قاعدة البيانات باستخدام أوامر مثل: GET...@.

الوضع البديل لهذه الدالة هو لا (OFF) ويعني أن حقول الادخال في الأوامر السابقة ستظهر بنفس اللون المستخدم للكتابة على الشاشة.

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» التعبير (<*expL*>) .

مثـــال:

تسمح لك الأوامر التالية بإدخال البيانات بدون إظهار ألوان للشاشة العكسية (Reverse Video) أو بحيث تكون ألوان حقول الادخال هي نفسها الألوان المستخدمة مع الشاشة.

```
RXP=SPACE(5)
SET INTENSITY OFF
SET DELIMITED ON
@ 12,10 SAY "Enter expression: " GET EXP
READ
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@...SAY – GET – SET COLOR

الأمر SET KEY

يخصص أحد المفاتيح لاستدعاء برنامج أو إجراء أثناء توقف البرنامج .

الشكل العام:

SET KEY (<*expN*> TO [Procedure]

حيث:

<*expL*> : رقم يمثل المفتاح الذي سيتم الضغط عليه .

(<*Procedure*>) : اسم البرنامج أو الاجراء الذي سينفذ عند الضغط على المفتاح .

الشرح:

يمكن تخصيص حتى ٣٢ مفتاحاً في البرنامج الواحد بحيث يخصص لكل مفتاح برنامج أو إجراء ينفذ أثناء توقف البرنامج مؤقتاً باستثناء مفتاح F1 لأنه دائماً مخصص لاستدعاء شاشات المساعدة. ويمكن أن يكون المفتاح مفتاحاً واحداً مثل F2 أو مفتاحان مثل Ctrl-F2 أو Alt-F2 والأوامر التي تسبب توقف البرنامج مؤقتاً هي :

WAIT – ACCEPT – READ – INPUT – MENU TO .

عندما يتم تنفيذ أمر SET KEY نتيجة ضغط المفتاح المخصص فإن الاجراء أو البرنامج المستدعى يتلقى ٣ قيم هي : اسم البرنامج المستدعي ورقم السطر الذي استدعى الاجراء واسم حقل الذاكرة الذي ينتظر الادخال.

يتشابه أمر SET KEY مع أمر SET FUNCTION إلا أن أمر SET KEY له أولوية التنفيذ عن أمر SET FUNCTION .

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» هذا الأمر.

مثال:

المثال التالي يستخدم برنامج KEY.PRG وفي هذا البرنامج فإن ضغط مفتاح

F10 أثناء توقف البرنامج استجابة لأمر READ يستدعى الاجراء COMPACC الذي يقوم بإظهار أسماء الشركات وأرقام الحسابات.

```
* Program: KEY.PRG
SET KEY -9 TO COMPACC       && تعنى مفتاح F10
CLEAR
DO WHILE .T.
  MCOMP=SPACE(3)
  MACCOUNT=SPACE(8)
  @ 10,5 SAY "Enter company name: " GET MCOMP
  @ 12,5 SAY "Enter account no. : " GET MACCOUNT
  READ                          && تسبب توقف البرنامج مؤقتا
  IF MCOMP = " "
     RETURN
  ENDIF
ENDDO
*
* سيتم تنفيذ الاجراء  التالى عندما يتوقف البرنامج مؤقتا ويضغط مفتاح F10
PROCEDURE COMPACC
PARAMETERS PROG,LINE,M_VAR  && فى هذا المثال سيقبل الاجراء  معطيات
                            && حتى ولو لم يتم تعريفها
OLDAREA = SELECT()          && ضع رقم المنطقة الحالية فى الحقل OLDAREA:
SELECT 0                    && اختر أول منطقة تالية
SAVE SCREEN                 && احفظ الشاشة الحالية داخل الذاكرة
@ 0,0 CLEAR                 && ننصح بعدم استخدام أمر CLEAR
                            && لأنه  قد يحذف الحقول الموجودة فى الذاكرة
DO CASE
   CASE UPPER(M_VAR) = "MCOMP"   && فى حالة الاستفسار  عن أسماء الشركات
      * <commands to list companies>
   CASE UPPER(M_VAR) = "MACCOUNT"   && فى حالة الاستفسار  عن الحسابات
      * <commands to list accounts>
ENDCASE
WAIT "Press a key"   && بعد عرض أسماء الشركات وأرقام الحسابات
                     && انتظر حتى يتم ضغط أحد المفاتيح
                     && ثم استرجع الشاشة المحفوظة واستأنف التنفيذ
RESTORE SCREEN
SELECT OLDAREA                  && ارجع الى المنطقة الأصلية
RETURN
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

KEYBOARD - SET FUNCTION - LASTKEY( )

الأمـر SET MARGIN

يحدد الهامش الأيسر للطابعة.

الشكل العام:

SET MARGIN TO <*expN*>

حيــث:

<*expN*> : عدد مسافات الهامش الأيسر.

الشــرح:

يستخـدم هذا الأمر لتحديد الهامش الأيسر للطابعة. فإذا تم ذلك فإن عدد فراغات الهامش الأيسر يكون مساويا للعدد المذكور في الأمر. وإذا لم تحدد للطابعة الهامش الأيسر فإن قاعدة البيانات تخصص للهامش قيمة قدرها صفر.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـال:

الأمر التالي يخصص الهامش الأيسر للطابعة بعشرة مسافات.

SET MARGIN TO 10

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

SET PRINT

الأمر SET MESSAGE

يحدد المكان الذي ستظهر فيه رسالة المحث الموجودة في أمر @...PROMPT

الشكل العام:

SET MESSAGE TO [<*expN*> [CENTER]]

حيث:

<*expN*>	:	رقم السطر الذي ستظهر عليه الرسالة.
[CENTER]	:	لضبط الرسالة وسط السطر.

الشرح:

يحدد أمر SET MESSAGE السطر الذي ستظهر عنده رسالة المحث الموجودة في أمر @...PROMPT . والأمر بدون معطيات يوقف إظهار الرسالة ويجب الانتباه إلى أن هذا الأمر لا يحذف الفراغات الموجودة قبل أو بعد الرسالة ولذلك فيفضل إعداد وظيفة خاصة أو إجراء لهذه العملية حتى لا تتداخل أجزاء الرسالة الطويلة مع الرسائل القصيرة.

الاختلاف عن *dBASE III PLUS* : أمر SET MESSAGE الموجود في «دي بيس» مختلف تماماً عن هذا الأمر.

مثـــال:

المثال التالي يظهر رسالة "First choice" أو "Second choice" في وسط السطر الرابع والعشرين.

```
SET MESSAGE TO 24 CENTER
@ 10,5 PROMPT "1st" MESSAGE "First choice"
@ 12,5 PROMPT "2nd" MESSAGE "Second choice"
MENU TO ACTION
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

@...PROMPT – MENU TO – SET WRAP

الأمـــر SET ORDER

يحدد أيًّا من ملفات الفهرسة المفتوحة يجب أن يكون هو الرئيسي.

الشكل العام:

SET ORDER TO [<expN>]

حيـــث:

<expN> : رقم ملف الفهرس المطلوب أن يكون هو الرئيسي.

الشـــرح:

عندما تصدر أمر USE...INDEX أو أمر SET INDEX TO فإن أول ملف يذكر اسمـه بعد كلمة INDEX في الأمر الأول أو بعد كلمة TO في الأمر الثاني يعتبر هو الـرئيسي أي هو الملف الـذي يؤخـذ في الاعتبـار عنـدما تصـدر أمـر SEEK أو FIND لتبحث في الملف المفهرس أو عندما تصدر أمر LIST لترى ترتيب سجلات الملف. ويكون رقمه بالنسبة لقاعدة البيانات ١ والملف الذي يذكر بعده رقمه ٢ . . . وهكذا.

ومعروف أنك تستطيع أن تفتح حتى ١٥ ملفاً مفهرساً في نفس الوقت. فإذا رغبت في تغيير ترتيب أولويات ملفات الفهرسة بدون إغلاقها أو إعادة فتحها فيمكنك استخدام أمر SET ORDER .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

يستخدم المثال التالي ملف قاعدة البيانات STOCK.dbf ومعه ملفان مفهرسان الأول IACCT.ndx كفهرس رئيسي (رقمه في هذه الحالة ١) والثاني ICOMP.ndx (رقمه في هذا الحالة ٢).

```
USE STOCK INDEX IACCT, ICOMP
```

لكي تجعل ملف ICOMP.ndx هو الرئيسي ليتم البحث فيه أو لتظهر سجلات الملف من خلاله استخدم الأمر التالي:

```
SET ORDER TO 2
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

INDEX – SET INDEX – INDEXORD() – INDEXKEY( )

الأمر SET PATH

يحدد الطريق الذي يجب أن تسلكه قاعدة البيانات عند البحث عن الملفات.

الشكل العام:

SET PATH TO [<*path list*>]/(<*expC*>)

حيث:

<*path list*>	:	أسماء الأدلة (Pathes) مفصولة بعلامة « , ».
<*expC*>	:	تعبير حرفي يشتمل على أسماء الأدلة.

الشرح:

عادة تبحث قاعدة البيانات عن الملفات الطلوبة (لفتحها - أو لحذفها أو لتغيير اسمها. . . الخ) في القرص والدليل المخصصين فإذا أردت أن توجه قاعدة البيانات لتبحث في دليل آخر غير الدليل الذي توجد عليه أو في قرص آخر غير المخصص معك عن الملفات المطلوبة استخدم هذا الأمر.

وكلمة Path معناها طريق فإذا كنت تريد لقاعدة البيانات أن تبحث عن الملفات في أكثر من طريق فيجب أن تحدد هذه الطرق بأولويات البحث ويفصل بين كل طريق وآخر بعلامة « , ».

ويجب الانتباه إلى أن أمر SET PATH يستخدم للبحث عن ملفات موجودة أما بالنسبة للأوامر التي تنشىء ملفا جديدا مثل USE - INDEX - TOTAL . . . الخ. فإذا كنت تريد وضعها على دليل آخر أو قرصا آخر فيجب أن تحدد اسم الدليل ومشغل القرص (Path) قبل اسم الملف.

إذا استخدمت هذا الأمر بصيغة SET PATH TO فقط بدون اختيارات أخرى فسيلغي كل الطرق (Pathes) المفتوحة وسيوجه قاعدة البيانات للبحث في الدليل المختار فقط.

الاختلاف عن dBASE III PLUS : لا تستخدم «دي بيس» التعبير (*<expC>*) .

مثـــال :

توجه الأوامر التالية قاعدة البيانات للبحث عن ملف برنامج اسمه MAS-TER.prg إذا لم تجده في الدليل والقرص المختارين في الطرق التالية بالترتيب التالي :

١ ـ تبحث في مشغل الوحدة D تحت دليل اسمه DBMS\EMP\ .

٢ ـ تبحث في مشغل الوحدة A تحت الدليل الرئيسي .

```
SET PATH TO D\DBAS\EMP, A:
DO MASTER
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

DIR – SET DEFAULT

الأمر SET PRINT

يوجه أو لا يوجه المخرجات إلى الطابعة .

الشكل العام:

SET PRINT on/OFF/(<*expL*>)

حيـــث :

(<*expL*>) : تعبير منطقي .T. أو .F. بدلا من ON أو OFF .

الشـــرح :

الـوضـع الأصـلي لهذه الـدالـة هو لا (OFF) . والـوضـع البـديل لها هو SET PRINT ON ويسمح بتوجيه المخرجات التي تظهر على الشاشة إلى الطابعة أيضا . إلا أنه لا يؤثر على مخرجات أمر SAY...@ . فإذا كنت تريد توجيه مخرجات الأمر SAY...@ إلى الطابعة فيجب أن تستخدم أمر SET DEVICE TO PRINT .

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» التعبير (<*expL*>) .

مثـــال :

تسمح الأوامر التالية بإظهار الأسهم وأسعارها على الشاشة والطابعة معا .

```
USE STOCK
SET PRINT ON
LIST PRICE,NO_SHARES,PRICE * NO_SHARES
SET PRINT OFF
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

SET DEVICE – EJECT

الأمـــر SET PRINTER

يخصص وحدة طباعة لاستقبال المخرجات.

الشكل العام:

SET PRINTER TO [<*device*>/<*file*>/(<*expC*>)]

حيـــث:

<*device*>	:	الاسم الدال على الطابعة.
<*file*>	:	اسم ملف نصي ستوضع عليه مخرجات الطابعة لحين طباعته.
(<*expC*>)	:	عبارة حرفية تدل على الطابعة.

الشـــرح:

عادة توجـد المخـرجـات إلى الطابعة المتصلة بالحاسب وتعرف الطابعة لنظام التشغيل باسم LPT1 أو LPT2 أو LPT3 أمـا إذا كانت الـطابعـة متصلة مع شبكـة اتصالات محلية (Serial printer) فتعرف باسم COM1 أو COM2 . . . وهكذا.

فإذا كانت عندك طابعة واحدة فلست في حاجة لاستخدام هذا الأمر أما إذا كنت تستخـدم أكثـر من طابعـة أو إذا كنت متصـلا مع شبكة اتصالات فيمكنك الاستعانة بهذا الأمر لتوجيه المخرجات إلى طابعة معينة. أو إلى ملف نصي لطباعته فيما بعد إذا استخدمت الأمر بدون معطيات يوجه المخرجات إلى الطابعة LPT1 .

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» اختيار <*file*> أو (<*expC*>).

مثـــال:

بفرض أن الطابعة المتصلة بحاسبك من نوع ابسون وبفرض أنها متصلة بالغرفة (Port) رقم ١، وبفرض أن عندك طابعة ليزر متصلة بالغرفة (Port) رقم ٢ وتريد طباعة التقرير STKRPRT.frm على طابعة الليزر فيمكنك استخدام الأوامر التالية:

```
SET PRINTER TO LPT2
USE STOCK INDEX ICOMP
REPORT FORM STKRPRT TO PRINT
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET PRINT – SET DEVICE

الأمر SET PROCEDURE

يعرف ملف يشتمل على برامج صغيرة أو إجراءات أو وظائف خاصة لتتم ترجمتها ضمن النظام.

الشكل العام:

SET PROCEDURE TO [<*procedure file*>]

حيث:

<*procedure file*> : اسم الملف الذي يشتمل على البرامج الصغيرة أو الوظائف الخاصة.

الشرح:

يمكن وضع أي عدد من البرامج الصغيرة أو الوظائف الخاصة تسمح به الذاكرة داخل ملف واحد ويتم ترجمة (Compiling) هذه البرامج والوظائف ضمن النظام. فعندما يكتشف «كلبر» أمر PROCEDURE أو FUNCTION يقوم بترجمة الأوامر التالية له حتى يصل إلى أمر RETURN أو إلى أمر يوجهه أن الاجراء انتهى مثل وجود أمر PROCEDURE أو FUNCTION جديد.

الاختلاف عن *dBASE III PLUS* : يختلف هذا الأمر في «كلبر» عنه في «دي بيس» فمثلاً لا يلزم «كلبر» فتح ملف الاجراءات وغلقه. وإنما يستخدم هذا الأمر في «كلبر» مثل أمر INCLUDE في لغة C .

مثال:

المثال التالي يجعل جميع الاجراءات (PROCEDURES) والوظائف الخاصة (User Defined Functions) الموجودة في الملفات الثلاثة التالية متاحة لجميع البرامج الأخرى في النظام لتتعامل معها.

```
* Program: MAIN.PRG
SET PRODECURE TO PROC1
SET PRODECURE TO PROC2
SET PRODECURE TO PROC3
* < Other commands>
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

PROCEDURE – FUNCTION – DO

الأمر SET RELATION

يربط ملفي قاعدة بيانات طبقا لعلاقة مشتركة بينهما.

الشكل العام:

SET RELATION TO [<*key expression1*>/RECNO( )/<*expN1*> INTO <*alias1*>] [TO <*Key expression2*>/<*RECNO( )*>/<*expN2*> INTO <*alias2*>] ...[ADDITIVE]

حيث:

<*key expression*>	:	الحقل المشترك في الملفين والذي تمت فهرسة أحد الملفين طبقا لبياناته.
RECNO()	:	تربط الملفين بناء على رقم السجل.
<*expN*>	:	تعبير رقمي يستخدم الوظيفة RECNO()
<*alias*>	:	اسم الملف المطلوب ربطه مع الملف المفتوح أو رقم المنطقة الموجود بها.
[ADDITIVE]	:	تربط ملف غير موجود بسلسلة الملفات التي تم ربطها.

الشـرح:

يستخدم هذا الأمر لربط ملف قاعدة البيانات المفتوح بملف أو بملفات آخر في منطقة ثانوية (foreground area) بالذاكرة. ويتم ربط الملفات إما بناء على حقل مشترك بينهما يذكر بعد كلمة TO في الأمر [الاختيار <*key expression*>] أو بناء على رقم السجل [الاختيار RECNO()] فيتم ربط السجل الأول في الملف الأول بالسجل الأول في الملف الثاني والسجل الثاني بالملف الأول بالسجل الثاني في الملف الثاني. . . وهكذا.

ويجب أن يكون الملف المزمع ربطه مع ملف قاعدة البيانات سبق فهرسته طبقا لبيانات الحقل المشترك والذي يعوض عنه في الأمر بعبارة *<key expression>* وذلك لأنه بمجرد ربط الملفين أو بعد انتهاء تنفيذ الأمر فإن تحريك المؤشر داخل ملف قاعدة البيانات المختار (the current selected database) يتسبب في نقل المؤشر إلى السجل المقابل في الملف الآخر المربوط معه.

ويتم التعرف على السجل المقابل بمطابقة محتويات الحقل المشترك في الملفين. فإذا وجدت قاعدة البيانات قيمة حقل في الملف الأول غير موجودة بالملف الثاني فإنها تعتبر أنها وصلت إلى نهاية الملف. وبالتالي تصير الوظيفة ()EOF صحيحة (.T.) .

لفض العلاقات الموجودة بين الملفات استخدم الأمر بصيغة SET RELATION TO فقط. يمكن ربط حتى ٨ ملفات فرعية مع الملف الأصلي.

الاختلاف عن dBASE III PLUS : تربط «دي بيس» ملف أصلي مع ملف واحد فقط في حين يستطيع «كلبر» ربط ٨ ملفات مع ملف أصلي.

مثـــال:

المثال الآتي يربط ملفات STUDENTS.dbf و COURSES.dbf و TEACHERS.dbf الموجودين طبقاً لبيانات حقل STUDENTNO .

لوحظ أن تحريك المؤشر داخل ملف STUDENTS.DBF سيحرك المؤشر داخل ملفات COURSES.DBF و TEACHERS.DBF لنفس السجل الذي يحمل نفس رقم الطالب (STUDENTNO) .

```
SELEC A
USE STUDENTS
SELECT B
USE COURSES INDEX ST_NO
SELECT C
USE TEACHERS INDEX ST_NO
SELECT A
SET RELATION TO ST_NO INTO COURSES, TO ST_NO INTO TEACHERS
```

لكي تظهر بيانات الطالب في الملفات الثلاثة استخدم الأمر التالي

```
LIST STUDENTNO, LASTNAME, B->COURSE1, B->COURSE2, C->TEACH1
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

INDEX – SET INDEX – SET ORDER

الأمر SET SCOREBOARD

يظهر أو يلغي الرسالة التي تظهر في السطر رقم صفر مع أمر READ أو MEMOEDIT() .

الشكل العام:

SET SCOREBOARD ON/off/(<*expL*>)

حيث:

(<*expL*>) : تعبير منطقي .T. أو .F. بدلاً من ON و OFF .

الشرح:

الوضع الأصلي لهذه الدالة هو نعم (ON) . ويعني إظهار رسالة في السطر الأول من الشاشة (السطر رقم صفر) من قبل قاعدة البيانات لتوضح لك بعض المعلومات مثل Caps للدلالة على أن مفتاح Caps تم الضغط عليه أو الرسالة التي تظهرها قاعدة البيانات عند استخدام الوظيفة MEMOEDIT() .

الوضع البديل لهذه الدالة هو لا (OFF) ويعني إلغاء مثل هذه المؤشرات التي تظهر في السطر الأول من الشاشة.

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» التعبير (<*expL*>) .

مثال:

لتمنع ظهور مثل هذه المؤشرات على الشاشة استخدم الأمر التالي:

```
SET SCOREBOARD OFF
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

READ – MEMOEDIT()

الأمـــر SET SOFTSEEK

تحدد مكان وضع المؤشر داخل الملف بعد البحث بأمر SEEK .

الشكل العام:

SET SOFTSEEK on/OFF/(<*expL*>)

حيـــث:

(<*expL*>) : تعبير منطقي .T. أو .F. بدلاً من ON و OFF .

الشـــرح:

الوضع الأصلي لهذا الأمر هو OFF . ومعناه أن المؤشر يوضع بعد آخر سجل بالملف إذا لم تجد قاعدة البيانات السجل الذي تبحث عنه وبالتالي تصير الوظيفة EOF( ) صحيحة (.T.) والوظيفة FOUND( ) خاطئة (.F.) .

أما الوضع البديل الذي يمكن الانتقال إليه فهو ONومعناه أن قاعدة البيانات إذا لم تجد السجل الذي تبحث عنه داخل الملف فإنها تضع المؤشر عند أول سجل يلي السجـل المـطلـوب وبالطبع ستكون قيمته أعلى من المطلوب وبالتالي فـإن الوظيفـة EOF( )ستبقى خاطئة (.F.) والوظيفة FOUND( )أيضا خاطئة (.F.) . فمثلاً إذا كنت تبحث عن الرقم ١٤٠,٥٠٠ في حقل السعر وكان هذا الرقم غير موجود بالملف والرقم التـــالي له هو ١٤٤,٥٠٠ فإن المؤشر سيقف عنـــد السجـــل الـذي يحمـل الـرقم ١٤٤,٥٠٠، إذا كان الأمر في حالة ON . أما إذا كان الأمر في حالة OFF فسيقف المؤشر بعد آخر سجل.

الاختلاف عن *dBASE III PLUS* : الأمر غير موجود بها.

مثــــال :

```
USE STOCK INDEX IPRICE
SEEK 140.500                 && هذا السعر غير موجود بالملف &&
? FOUND() , EOF()            && .F. , .T.  الاجابة
SET SOFSEEK ON
SEEK 140.500
? FOUND() , EOF()            && .F. , .F.  الاجابة:
? PRICE                      && 144.500   الاجابة:
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SEEK – SET EXACT

الأمر SET TYPEAHEAD

يحدد حجم محطات التخزين الانتقالية.

الشكل العام:

SET TYPEAHEAD TO <*expN*>

حيث:

<*expN*> : عدد الحروف التي ستخزن في المحطة.

الشرح:

الأوامر التي تدخل إلى قاعدة البيانات تخزن مؤقتا في محطة تخزين مؤقتة يطلق عليها (Tpyeahead buffer) لحين إنتهاء قاعدة البيانات من تنفيذ الأمر الذي يشغلها ثم تقرأ الأمر الجديد عندما تكون جاهزة. هذه المحطة لها سعة محددة قدرها ١٦ حرفا. ويستخدم هذا الأمر لتحديد الحروف التي يمكن تخزينها بهذه المحطة المؤقتة (Typeahead buffer) . فقد تحتاج لزيادة عدد الحروف التي تريد تخزينها مؤقتا بهذا المكان أو لتقليلها. وعدد الحروف المسموح بوضعها في هذه المحطة يتراوح بين صفر و٣٢,٧٦٨.

لاحظ أن تخصيص صفر لهذه المحطة يوقف كلا من أمر Alt-C و Alt-D .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة:

INDEY() – SET BELL – CLEAR TYPEAHEAD

الأمر SET UNIQUE

يحدد هل يظهر كل السجلات المتشابهة في ملفات الفهارس أم يظهر السجل الأول فقط من كل مجموعة.

الشكل العام:

SET UNIQUE on/OFF/(<*expL*>)

حيث:

(<*expL*>) : تعبير منطقي .T. أو .F. بدلاً من ON و OFF .

الشرح:

الوضع الأصلي لهذه الدالة هو لا (OFF) والوضع البديل لها هو نعم (ON) ويعني حذف السجلات المكررة التي تحمل نفس البيانات داخل الحقل المختار. (key field) عندما تنشىء ملف فهرس جديد بعبارة أخرى فإن ملف الفهرس الجديد سيشتمل على السجل الأول من كل مجموعة سجلات متشابهة.

وفي حالة الحاجة لاعادة فهرسة ملف أنشىء بهذه الطريقة فسيظل الملف محتفظا بوضعه الأصلي عند إنشائه أي سيظل يشتمل على سجل واحد من كل مجموعة سجلات متشابهة. وبالمثل يظل الملف المفهرس محتفظا بوضعه الأصلي في حالة إضافة أو حذف سجلات إلى أو من الملف الأصلي.

الاختلاف عن *dBASE III PLUS*: باستخدام «كلبر» كل الملفات تكون في حالة واحدة. إما UNIQUE أو لا، يمكن باستخدام «دي بيس» جعل كل ملف في حالة مختلفة عن الآخر.

مثـال:

المثال الآتي ينشىء ملفا مفهرسا باسم ICOMP.ndx بناء على بيانات حقل COMPANY ولكن بشرط ألا تتكرر السجلات المتشابهة في الحقل.

```
USE STOCK
SET UNIQUE ON
INDEX ON COMPANY TO ICOMP
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

INDEX – REINDEX – SET INDEX – USE – FIND – SEEK

الأمــر SET WRAP

يسمـح أو يمنـع الانتقـال من الاختيار الأخـير إلى الاختيار الأول في حالـة استخدام القوائم ذات الشريط المضاء.

الشكل العام:

SET WRAP on/OFF/(<*expL*>)

حيـــث:

(<*expL*>) : تعبير منطقي .T. أو .F. بدلاً من ON و OFF .

الشـــرح:

إذا أردت أن تجعل اختيارات القوائم التي تستخدم الأشرطة المضاءة تلف حول بعضهـا بعبـارة أخرى إذا وصلت إلى آخر اختيار وضغطت مفتاح السهم ↓ فينقل الشريط المضاء إلى الاختيار الأول في القائمة، ضع هذا الأمر في حالة ON .

الاختلاف عن *dBASE III PLUS* : لا يوجد بها.

مثـــال:

```
SET WRAP ON                 &&  اسمح للشريط المضاء بالحركة من اسفل لاعلى
SET MESSGAE TO 1            &&  اختر سطر رقم لاظهار الرسالة الاختيارية
@ 10,05 PROMPT " Maintenance    " MESSAGE " Add, Delete, Edit    "
@ 12,05 PROMPT " Query          " MESSAGE " Ask questions        "
@ 14,05 PROMPT " Reports        " MESSAGE " Reports Menu         "
@ 16,05 PROMPT " Exit           " MESSAGE " Exit to DOS          "
MENU TO ACTION
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@...PROMPT – MENU TO

الأمر SKIP

يحرك المؤشر داخل الملف للأمام أو للخلف لعدد محدد من السجلات.

الشكل العام:

SKIP [<*expN1*>] [ALIAS <*workarea*>/<*alias*>/(<*expN2*>)]

حيث:

<*expN1*> : تسمح بتحريك المؤشر لعدد من السجلات مساو لقيمتها.

ALIAS <*workarea*>/<*alias*>/(<*expN2*>)

: تعبير يحدد اسم المنطقة التي سيتم البحث فيها.

الشرح:

يسمح هذا الأمر بتحريك المؤشر داخل الملف المفتوح للأمام أو للخلف في المنطقة المحددة بالرقم أو الاسم. فإذا أصدر الأمر بدون <*expN1*> فإن المؤشر ينتقل تلقائيا إلى السجل التالي لموقعه. وعادة يتحرك المؤشر للأمام إلا إذا اشتمل التعبير الرقمي على علامة ـ فإنه يتحرك إلى الخلف.

إذا كان المؤشر عند أول سجل بالملف وأصدر أمر SKIP -1 فإن الوظيفة RECNO( ) تظل تعطي الرقم ١ وتصبح الوظيفة BOF( ) صحيحة (.T.).

أما إذا كان المؤشر عند آخر سجل بالملف وأصدر أمر SKIP فإن الوظيفة RECNO( ) تشتمل على رقم آخر سجل زائدا واحدا وتصبح الوظيفة EOF( ) صحيحة (.T.).

إذا كان الملف مفهرسا فإن أمر SKIP ينقل المؤشر حسب ترتيب الملف المفهرس أما إذا كان غير مفهرس فإن المؤشر ينتقل حسب ترتيب أرقام السجلات.

الاختلاف عن *dBASE III PLUS*: لا تستطيع «دي بيس» تحريك المؤشر في منطقة غير المنطقة المختارة (Current work area).

مثـــال :

```
USE STOCK
?RECNO()                  && 1   الإجابة
?BOF()                    && .F.  الإجابة
SKIP -1
?BOF()                    && .T.  الإجابة
SKIP 2 IN ALIAS B         && 2 اقفز سجلين فى المنطقة رقم
                          && 1 بافتراض أن المنطقة الحالية هى رقم
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

COMMIT – GO – BOF( ) – EOF( )

الأمر SORT

ينشيء ملفا جديدا يشتمل على سجلات ملف آخر مرتبة طبقا لورودها في حقل معين.

الشكل العام:

SORT TO <*new file*> ON <*field*> [/A] [/C] [/D] [, <*field2*> [/A] [/C] [/D]...]

[<*scope*>] [FOR <*condition*>] [WHILE <*condition*>]

حيث:

<new file>	: اسم الملف الذي سيحتوي على السجلات بعد ترتيبها.
<field>	: الحقل الذي سترتب السجلات في الملف الجديد طبقا لبياناته.
/A	: تجعل الفهرس يتم تصاعديا.
/C	: تجعل قاعدة البيانات لا تفرق بين الحروف الكبيرة والصغيرة (Upper and Lower Case Letters)
/D	: تجعل الفرز يتم تنازليا.
<Scope>	: تحدد السجلات التي سيتم فرزها مثل All, Rest .
FOR <Condition>	: تجعل الفرز ينفذ فقط على السجلات التي تتطابق مع الشرط الموجود في الأمر.
WHILE <Condition>	: تسمح باستمرار الفرز طالما أن الشرط الموجود في الأمر صحيحا.

الشــــرح:

يستخدم هذا الأمر لفرز أو ترتيب سجلات ملف قاعدة البيانات المفتوح طبقا لبيانات الحقل أو الحقول المختارة (<field> أو <field2>) ويضع السجلات الجديدة

بعد ترتيبها على ملف جديد (<new file>) . وعادة يتم ترتيب السجلات في الحقول الحرفية تصاعديا ما لم يحدد في الأمر الاختيار "D/". أما الحقول الرقمية والتاريخية فيتم فرزها من الأصغر إلى الأكبر.

الحقول المنطقية وحقول الملاحظات لا تستخدم أساساً للفرز.

الملف الجديد الذي ينشأ نتيجة تنفيذ هذا الأمر يأخذ اسما ممتدا هو "dbf." ولذلك لكي ترى السجلات الجديدة بعد عملية الفرز يجب أن تفتح الملف أولا بواسطة أمر USE ثم تستخدم أحد أوامر الاظهار المعروفة .

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف .

مثـــال :

المثال الآتي يستخدم ملف STOCK.dbf ليرتب السجلات الخاصة بشركة IBM فقط بدون تفرقة بين الحروف الكبيرة والصغيرة. ويضع السجلات الجديدة المرتبة على ملف جديد اسمه SORTFILE.dbf .

```
SORT TO SORTFILE ON ACCOUNTNO/AC FOR COMPANY = "IBM"
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

INDEX – ASORT( )

الأمر STORE

يستخدم لتخزين تعبير ما داخل حقل ذاكرة.

الشكل العام:

STORE <*expression*> TO <*memvar list*>

memvar = <*expression*>

حيث:

<*expression*>	:	قد تكون تعبيرا حرفيا أو رقميا أو تاريخيا أو منطقيا.
<*memvar list*>	:	اسم حقول الذاكرة التي سيخزن بها التعبير (<*Expression*>).
<*memvar*>	:	اسم حقل الذاكرة.

الشرح:

يخزن هذا الأمر التعبير الموجود في <*expression*> في كل من حقول الذاكرة المذكورة بعد كلمة TO .

إذا كانت هناك حقول بالذاكرة موجودة من قبل ومعرفة بنفس الاسم فإن القيم الجديدة لحقول الذاكرة المعرفة بنفس الاسم تحل محل القيم القديمة.

الصيغة البديلة لأمر STORE هي استخدام علامة = وتستخدم لتخصيص تعبير ما لحقل ذاكرة واحد وتستخدم بهذه الصورة:

<*memvar*> = <expression>

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

أمثلـــة :

١ - لوضع الرقم صفر في كل من حقول الذاكرة TOTAL1 - TOTAL2 - TOTAL3 استخدم الأمر التالي :

```
STORE 0 TO TOTAL1, TOTAL2, TOTAL3
```

٢ - لوضع العبارة "Student's Name:" في حقل ذاكرة اسمه M_Name استخدم الأمر التالي :

```
M_NAME = "Student's Name:"
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

CLEAR MEMORY - PRIVATE - PUBLIC - RELEASE

الأمر SUM

يجمع الحقول الرقمية في الملف المفتوح.

الشكل العام:

SUM [<*expN list*>] [<*scope*>] [FOR <*condition*>
[WHILE <*condition*>] [TO <*memvar list*>]

حيث:

<*scope*>	:	يحدد السجلات التي سيتم تجميعها من الملف.
<*expN list*>	:	أسماء الحقول الرقمية المطلوب تجميعها.
FOR <*condition*>	:	تسمح بتجميع السجلات التي تتطابق مع الشرط المذكور في الأمر.
WHILE <*condition*>	:	تسمح بتجميع السجلات طالما أن الشرط المذكور في الأمر صحيحا.
TO <*memvar list*>	:	أسماء حقول الذاكرة التي ستخزن بها النتائج.

الشـــرح:

يستخدم هذا الأمر لايجاد حاصل جمع الحقول الرقمية في الملف المفتوح. إذا استخدم الأمر بدون أية اختيارات إضافية فإن جميع الحقول الرقمية الموجودة في الملف يتم تجميعها أما إذا اشتمل الأمر على اسم أو أسماء بعض الحقول فإن الحقول المذكورة أسماءها فقط هي التي يتم تجمعيها. وكذلك إذا اشتمل الأمر على احد الاختيارات <*scope*> أو FOR/WHILE فإن السجلات المتطابقة مع الشرط الموجود في الأمر هي التي يتم تجميعها.

إذا أردت تخزين النتائج التي تحصل عليها من هذا الأمر فيجب أن تستخدم الاختيار TO <*memvar list*>
ثم تذكر أسماء حقول الذاكرة التي ستشتمل على النتائج.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــال :

المثال الآتي يستخدم ملف STOCK.dbf لايجاد حاصل جمع الحقل NO_SHARES للسجلات التي تتطابق بياناتها مع كلمة IBM في حقل COMPANY وتخزين حاصل الجمع في حقل ذاكرة اسمه M_SHARES .

```
USE STOCK
SUM NO_SHARES FOR COMPANY = "IBM" TO M_SHARES
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

AVERAGE – COUNT – TOTAL

الأمر TEXT

يظهر أو يطبع نص داخل برنامج .

الشكل العام:

TEXT [TO PRINT] [TO FILE <*file*>]

<*text characters*>

ENDTEXT

حيث:

<*text characters*> : أي نص مطلوب إظهاره أو طباعته .

<*file*> : الملف النصي الذي سيتلقى المخرجات (.TXT)

الشرح:

يستخدم هذا الأمر لاظهار أو طباعة نص من مجموعة حروف أو سطور إما على الشاشة أو على الطابعة ويظهر النص الموجود بين كلمة TEXT وكلمة ENDTEXT كما هو موجود داخل البرنامج . ويجوز توجيه المخرجات إلى ملف نصي (.TXT) .

الاختلاف عن dBASE III PLUS : لا تستطيع «دي بيس» توجيه المخرجات إلى الطابعة أو ملف نصي .

مثال:

المثال الآتي يظهر النص المكتوب بين كلمتي TEXT وENDTEXT على الشاشة أو الطابعة كما هو.

```
TEXT
  Clipper is the best program to develop and compile database
     management systems.
ENDTEXT
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

?/?? - @...SAY - MLCOUNT( ) - MEMOLINE( )

الأمر TOTAL

ينشىء ملف قاعدة بيانات جديد ويضع عليه إجماليات مختصرة لملف آخر سبق فرزه أو فهرسته.

الشكل العام:

TOTAL ON <*key field*> TO <*new file*> [<*scope*>] [FIELDS <*field list*>] [FOR <*condition*>] [WHILE <*condition*>]

حيث:

<key filed>	:	الحقل الذي سيتم تجميع السجلات طبقا لبياناته.
<new file>	:	اسم الملف الجديد الذي ستوضع عليه الاجماليات المختصرة.
<scope>	:	يحدد السجلات التي سيتم تنفيذ الأمر عليها مثل ALL أو REST . . . الخ.
FIELDS <field list>	:	أسماء الحقول التي سيتم تجميعها.
FOR <condition>	:	تجعل التجميع يتم على السجلات التي تتطابق مع الشرط الموجود في الأمر فقط.
WHILE <condition>	:	تسمح باستمرار التجميع طالما أن الشرط الموجود في الأمر صحيحا.

الشـرح:

يجمع هذا الأمر الحقول الرقمية في الملف المفتوح بشرط أن يكون الملف المفتوح سبق فرزه أو فهرسته، فيأخذ سجلا واحدا لكل مجموعة سجلات تشتمل على نفس البيانات ويضع عليه إجمالي الحقول الرقمية لهذه السجلات ثم يضعه على الملف الجديد <new file> . وتكون باقي حقول هذا السجل هي نفسها الحقول التي يشتمل عليها أول سجل في المجموعة داخل ملف قاعدة البيانات الأصلي باستثناء حقل الملاحظات

memo field فإنه لا ينسخ إلى الملف الجديد. يأخذ الملف الجديد اسماً ممتداً هو ".dbf".

إذا اشتمل الأمر على الاختيار <field list> FIELDS فإن الحقول المحددة في الأمر فقط هي التي يتم تجميعها. إذا اشتمل الأمر على أحد الاختيارين <scopes> أو FOR/WHILE <conditions> فإن السجلات المتطابقة فقط هي التي يتم تجميعها.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

يجمع المثال الآتي الحقول الرقمية في ملف STOCK.dbf ويضع الاجماليات على ملف جديد اسمه TOTCOMP.

```
USE STOCK INDEX ICOMP
TOTAL ON COMPANY TO TOTCOMPANY
USE TOTCOMPANY
LIST
```

ستحصل على النتيجة التالية:

Record#	ACCOUNTNO	TRANSID	COMPANY	TYPE	DATE	NO_SHARES	PRICE
1	066882	002	ATT	B	01/01/86	160	207.875
2	014786	001	IBM	B	05/01/86	235	455.000

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة:

SUM – AVERAGE

الأمر TYPE

يظهر محتويات ملف.

الشكل العام:

TPYE <*file.extension*> [TO PRINT] [TO FILE <*file*>]

حيث:

<file.extension>	:	اسم الملف المطلوب إظهار محتوياته.
TO PRINT/TO FILE	:	يرسل الناتج إلى الطابعة أو إلى الملف المذكور.

الشرح:

يظهر هذا الأمر محتويات الملفات النصية المكتوبة بشفرة.ASCII مثل TXT. أو PRG. أو FMT. فقط أما الملفات غير المكتوبة بشفرة ASCII مثل DBF. أو NDX. فلا يظهرها. يجب الانتباه إلى أن الملفات المفتوحة لا يمكن إظهار محتوياتها.

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» الاختيار TO FILE

مثال:

لاظهار محتويات البرنامج MAIN.prg على الطابعة أدخل الأمر الآتي:

TYPE MAIN. PRG TO PRINT

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET PRINTER TO

الأمــر UNLOCK

يفتح ملف أو سجل أغلق داخل النظام.

الشكل العام:

UNLOCK [ALL]

الشـــرح:

في حالة استخدام شبكة اتصالات محلية قد تحتاج لغلق ملف أو سجل لتمنع الآخرين من الكتابة عليه ويقوم هذا الأمر بإعادة الملف أو السجل إلى وضعه الأصلي. إذا استخدم الاختيار ALL فإن كل السجلات أو الملفات المغلقة عن الآخرين تعود إلى وضعها الأصلي وإذا لم يستخدم فإن الأمر ينصرف فقط إلى المنطقة المختارة.

تحريك المؤشر إلى سجـل جديد وإصـدار الوظيفة RLOCK( ) يجعل السجل متاحاً (يلغي تأثير UNLOCK).

الاختلاف عن *dBASE III PLUS* : غير موجود بها.

مثـــال:

```
IF RLOCK()                          && لو كان الملف مغلقا
   REPLACE PRICE WITH PRICE*1.10
   UNLOCK                           && افتح الملف
ELSE
   ? " Can't updtae... file locked "
   * < other commands>
ENDIF
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET EXCLUSIVE – USE – EXCLUSIVE – FLOCK( ) – RLOCK( )

الأمـــر UPDATE

يستخدم محتويات ملف معين لتعديل محتويات ملف آخر.

الشكل العام:

UPDATE ON <*key field*> FROM <*alias*>

REPLACE <*field*> WITH <*expression*> [FIELD2, WITH <*exp2...*>]

[RANDOM]

حيـــث:

<key field>	:	اسم الحقل المشترك بين الملفين.
<alias>	:	اسم لملف موجـود على القـرص يستخدم في عملية التعديل أو رمز يدل عليه.
<field>	:	اسم الحقل الذي ستتغير محتوياته.
<field2>/<exp2>..	:	تستخدم إذا كنت ستغير محتويات أكثر من حقل.
FOR <condition>	:	تعدل بيانات السجلات المتطابقة مع الشرط الموجود في الأمر.
WHILE <condition>	:	تسمح بالتعديل في السجلات التي تتطابق مع الشرط الموجود في الأمر طالما أن الحالة صحيحة.

الشـــرح:

يستخدم هذا الأمر بيانات ملف موجود على القرص (<alias>) لتعديل بيانات الملف المفتـوح (Active Database file) وذلك بمقارنة سجلات الملفين طبقا لبيانات حقل مشترك بينهما (Key field) . ويجب أن يكون الملف المشار إليه بالعبارة <alias> مفتوحا أيضا في واحدة من المناطق غير المختارة (Unselected Workarea) .

ويجب الانتباه إلى ضرورة فرز أو فهرسة السجلات التي ستتم مطابقتها قبل عملية التعديل. أما إذا كان الملف غير المختار والمشار إليه بالعبارة (<alias>) غير مفهرس أو غير مفروز على نفس الحقل المشترك في الملفين (<key field>) فيجب إضافة كلمة RANDOM إلى الأمر. في حالة استخدام شبكة اتصالات محلية يجب استخدام الملف الذي سيتم تعديله منفرداً (...Exclussrly...) .

الاختلاف عن *dBASE III PLUS* : تسمح «دي بيس» بالتعديل طبقا لمحتويات حقل واحد فقط وليس طبقا لتعبير يشمل أكثر من حقل.

مثـــال:

المثال الآتي يستخدم بيانات ملفي INVENT.dbf وPURCH.dbf لتغيير محتويات حقل الكمية الموجودة بالمخازن (ON_HAND) في ملف INVENT.dbf لتعكس الكمية الجـديدة بعـد عملية الشراء بعـد إضـافة الكمية المشتراة (QTY_RECV) من ملف المشتريات (PURCH.dbf) .

```
SELECT 2
USE PURCH INDEX PURCHNO
SELECT 1
USE INVENT INDEX INVNO
UPDATE ON STOCK_NO FROM PURCH REPLACE ON_HAND WITH ON_HAND+B->QTY_RECV
```

وتـلاحظ في هذا المثـال: أن الـرمـز B-> يستخـدم للاشـارة إلى ملف موجـود على القـرص ومختـار سابقـا في منـطقـة غير المـوجـود فيهـا الملف الحـالي (Current Active Database File) .

بعد تنفيذ أمر UPDATE فإن الكمية الموجودة بالمخازن (ON_HAND) في ملف (INVENT.dbf) ستزداد بمقدار الكمية الموجودة في حقل (QTY_RECV) الموجود في

ملف (PURCH.dbf) المختار سابقا فى المنطقة رقم ٢

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

REPLACE – SET RELATION – SELECT

الأمـر USE

يفتح ملف قاعدة البيانات والفهارس المذكورة في الأمر في المنطقة المختارة.

الشكل العام:

USE [<*database file*>/(<*expC*>)] [INDEX <*index file list*>] (<*expC*>)
[EXCLUSIVE] [ALIAS <*alias*>/(<*expC*>)

حيـث:

<database file>/(<*expC*>)	:	اسم ملف قاعدة البيانات المطلوب فتحه.
INDEX <index file list>	:	تفتح حتى ١٥ ملفاً مفهرساً.
ALIAS <alias>/<*expC*>)	:	اسم بديل للملف قاعـدة البيانـات في حالـة استخدام أكثر من ملف.
(EXCLUSIVE)	:	تمنع استخدام الملف بواسطة الآخرين.

الشـرح:

يستخدم هذا الأمر لفتح ملف قاعدة البيانات في المنطقة المحددة في أمر SELECT أو في المنطقة رقم ١ في حالة فتح ملف واحد فقط. إذا استخدم الأمر بدون أية اختيارات أخرى فإنه يغلق الملف المفتوح في المنطقة المختارة.

إذا استخدم معه الاختيار INDEX فإنه يفتح حتى ١٥ ملفاً سبق فهرستها باستخدام أمر INDEX . وتكتب أسماء ملفات الفهرسة مفصولة بعلامة «,» ويكون أول ملف بعد كلمة INDEX هو الملف الرئيسي (Master) الذي يتحكم في تنفيذ أوامر البحث في الملف المفهرس مثل FIND أو SEEK . ولكن ملفات الفهرسة الأخرى تتعدل بياناتها كلما أدخل سجلا جديدا للملف الأصلي حسب وضع الفهرسة الذي هي عليه.

الاختيار (EXCLUSIVE) يستخدم في حالة شبكة الاتصالات المحلية ليمنع

أي شخص آخر من التعامل مع الملف حتى يتم إغلاقه.

الاختيار alias قد يستخدم كاسم بديل لملف قاعدة البيانات بحيث يمكن التعامل مع محتويات الملف بالاسم الموجود بعد كلمة alias في حالة فتح أكثر من ملف في أكثر من منطقة.

الاختلاف عن dBASE III PLUS : يمكن فتح حتى ٧ ملفات فقط مع «دي بيس»، لا تستخدم «دي بيس» (<*expC*>) ولا [EXCLUSIVE] .

مثـــال :

بفرض انـك تريد فتح ملف قاعدة البيانات STUDENTS.dbf ومعه ملفان مفهرسان هما : STNO.ndx و STCITY.ndx. وتريد أن تجعل الاسم البديل لهذا الملف هو ST فيجب إدخال الأمر التالي :

```
USE STUDENTS INDEX STNO, STCITY ALIAS ST
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

CLOSE – INDEX – SELECT – SET INDEX

الأمـــر WAIT

يوقف تنفيذ البرنامج مؤقتا حتى يتم ضغط أحد المفاتيح .

الشكل العام:

WAIT [<*prompt*>] [TO <*memvar list*>]

حيـــث :

<*prompt*>	:	رسالة أو جملة تختارها لتظهر أثناء تنفيذ الأمر.
TO <*memvarC*>	:	اسم حقل الذاكرة الذي سيشتمل على الحرف الداخل من لوحة المفاتيح .

الشــــرح :

يوقف هذا الأمر تنفيذ البرنامج مؤقتا حتى يتم ضغط أحد مفاتيح لوحة المفاتيح فيستأنف تنفيذ البرنامج .

* إذا استخدم الأمر بصيغة WAIT فقط فإن الرسالة التلقائية "Press any key to continute" ستظهر قبل ضغط أحد المفاتيح .
* إذا اشتمل على رسالة (<*prompt*>) فإن هذه الرسالة ستظهر محل الرسالة التلقائية حتى يتم ضغط أحد المفاتيح .
* إذا أردت أن توقف تنفيذ البرنامج مؤقتا بدون إظهار الرسالة التلقائية استخدم الأمر بهذه الصورة :

WAIT " "

إذا أردت أن تخزن الحرف الذي سيدخل من لوحة المفاتيح في حقل ذاكرة أضف إلى الأمر الاختيار TO <*memvarC*> . وفي حالة ضغط مفتاح Enter ردا على أمر WAIT TO <*memvarC*> فإن حقل الذاكرة لن يحتوي على شيء .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــال :

المثال التالي يسأل المستخدم ما إذا كان يريد توجيه المخرجات إلى الطابعة. وفي حالة اختيار المستخدم الحرف Y ردا على أمر WAIT فإن جملة IF ستضع الطابعة في حالة نعم (ON) .

```
WAIT "Send the output to the printer? [Y/N]" TO YN
IF UPPER(YN) = "Y"
   SET PRINT ON
ENDIF
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

ACCEPT

الأمـر ZAP

يحذف جميع سجلات الملف المفتوح.

الشكل العام:

ZAP

الشـــرح:

يحذف هذا الأمـر جميع سجلات الملف المفتوح حذفا نهائيا. فهو مشابه تماما لاصدار أمري DELETE ALL ثم PACK.

ويتم تفريغ ملفات الفهرسة (.NDX) والملاحظات (.DBT) بمجرد تفريغ ملف .DBF .

الاختلاف عن *dBASE III PLUS* : لا تظهر «كلبر» رسالة تحذيرية قبل تفريغ الملف مثل «دي بيس» لأنه لا يستخدم أمر SET SAFETY .

مثـــال:

لحذف جميع سجلات ملف STUDENTS.dbf مرة واحدة استخدم الأوامر التالية:

```
USE STUDENTS
ZAP
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

DELETE – PACK

الفصل الرابع عشر

مرجع الوظائف

يشرح هذا الفصل بالتفصيل الوظائف الموجودة بمكتبات «كلبر». وقد أسهبنا في شرح الوظائف التي تتناول مفاهيم جديدة لم تكن موجودة في قاعدة البيانات dBASE III Plus والتي تمثل أهمية كبيرة لمستخدمي «كلبر» مثل وظائف القوائم والمصفوفات. أو تلك التي تمثل أهمية خاصة لأنها تقوم مقام أوامر غير موجودة في «كلبر» مثل وظيفة إظهار وتعديل البيانات DBEDIT( ) والتي جاءت بديلًا لأمر BROWSE في «دبيس» مع إدخال تحسينات كثيرة عليه. ووضعنا مع كل وظيفة مثالاً يشتمل على برنامج يعتبر نموذجاً وأفياً يمكن استخدامه ضمن تطبيقاتك التي تعدها في المستقبل.

الوظيفة ABS()

تعطى رقما مطلقا يميل الفرق بين رقمين بصرف النظر عن إشارة الرقم.

الشكل العام:

ABS(<*expN*>)

حيـــث:

<*expN*> : تمثل رقما أو تعبيرا رقميا.

الشـــرح:

تعطى هذه الوظيفة رقما يمثل الفرق بين رقمين أو عبارتين رقميتين بصرف النظر عن إشارة الرقم أي بصرف النظر عن أيهما أكبر.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

المثال التالي يوضح استخدام هذه الوظيفة لايجاد الفرق بين رقمين بصرف النظر عن أيهما أكبر.

```
STORE 20 TO VALUE1
STORE 10 TO VALUE2
? ABS(VALUE2 - VALUE1)          && 10  :الإجابة
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة:

الوظيفة ACHOICE( )

تظهر (تنفذ) قائمة تستخدم الشريط المضاء للانتقال بين اختياراتها. وتضع اختيارات هذه القائمة داخل مصفوفة.

الشكل العام:

ACHOICE(*<expN1>*, *<expN2>*, *<expN3>*, *<expN4>*, *<array1>*
[,*<array2>* [,*<expC>* [*<expN5>* [*<expN6>*]]]])

حيث:

<expN2> ... *<expN1>* : تمثل مكان الركن اليسار العلوي للقائمين.

<expN4> ... *<expN4>* : تمثل مكان الركن اليمين السفلي للقائمة.

<array1> : المصفوفة التي تشتمل على اختيارات القائمة.

<array2> : مصفوفة اختيارية ـ تشتمل على القيمة المنطقية .T. أو .F. وتستخدم .T. للاشارة إلى أن العنصر الذي يخصها داخل المصفوفة رقم ١ يمكن اختياره بينما تستخدم .F. للاشارة إلى أن العنصر الذي يخصها داخل المصفوفة رقم ١ لا يمكن اختياره.

<expC> : اسم للوظيفة الخاصة التي توجه استخدام الوظيفة ACHOICE( ).

<expN5> : رقم الاختيار داخل القائمة الذي سيوضع عنده الشريط المضاء في بداية تشغيل الوظيفة.

<expN6> : رقم السطر الذي سيوضع عنده الشريط المضاء في بداية تشغيل الوظيفة منسوباً إلى النافذة التي تظهر فيها القائمة.

الشـــرح:

تستخـدم الـوظيفـة ACHOICE( ) لانشـاء قائمة تشتمل على اختيارات يتم الانتقـال بينهـا باستخدام الشريط المضاء وتسمى هذه القائمة Pull-down menu . وتظهر هذه القائمة داخل نافذة. وتوضع اختيارات القائمة داخل مصفوفة (*<array1>*) وتشبه الوظيفة ACHOICE( ) الأمر @..PROMPT ...MENU TO

وتمتاز هذه الوظيفة على أمر MENU TO بميزتين:

الأولـــى: أن أمر MENU TO لا يسمح بأكثر من ٣٢ اختيار داخل القائمة.
الثانية: إذا كان حجم النافذة التي تشتمل على الإختيارات أقل من عدد الاختيارات فإن القائمة تطوى لأعلى أو لأسفل تلقائياً.

عندما يتم اختيار واحد من الاختيارات الموجودة بالقائمة فإن الوظيفة ACHOICE( ) تشتمل على رقم العنصر المختار. فإذا ألغيت القائمة باستخدام Esc فإن الوظيفة ستشتمل على الرقم صفر. ويجوز أن تشتمل الوظيفة ACHOICE( ) على اسم وظيفة خاصة (User Defined Function) لكي توجه استخدام بعض المفاتيح أثناء التنفيذ. وفي هذه الحالة فإن المفاتيح ستخصص لها وظيفة معينة. فإذا لم تشتمل على وظيفة خاصة فستخصص للمفاتيح وظائف تلقائية تسمى (Default Mode) .

ويوضـح الجـدول التـالي الـوظـائف المخصصة لبعض المفاتيح في حالة عدم استخدام وظيفة خاصة للتحكم في التنفيذ، وهو ما يطلق عليه Default Mode أي العمل بالطريقة التلقائية.

المفتاح	وظيفتـــه
↑	الانتقال إلى الاختيار السابق.
↓	الانتقال إلى الاختيار اللاحق.
Home	أول اختيار.
End	آخر اختيار.

المفتاح	وظيفتـــه
PgUp	الانتقال إلى الصفحة السابقة (منسوبة إلى حجم النافذة).
PgDn	الانتقال إلى الصفحة اللاحقة (منسوبة إلى حجم النافذة).
Ctrl-PgUp	أول اختيار.
Ctrl-PgDn	آخر اختيار.
Enter	تنفيذ الاختيار (يخصص رقم الاختيار للوظيفة ACHOICE())
Esc	إلغاء الاختيار (يخصص رقم صفر للوظيفة ACHOICE
←	الغاء القائمة (يخصص رقم صفر للوظيفة () ACHOICE)
→	الغاء القائمة (يخصص رقم صفر للوظيفة () ACHOICE)
أول حرف	تنفيذ الاختيار الذي يبدأ بالحرف

بينما يوضح الجدول التالي الوظائف المخصصة لبعض المفاتيح في حالة استخدام وظيفة خاصة للتحكم في التنفيذ.

المفتاح	وظيفتـــه
↑	الانتقال إلى الاختيار السابق.
↓	الانتقال إلى الاختيار اللاحق.
PgUp	الانتقال إلى الصفحة السابقة (منسوبة إلى حجم النافذة).

المفتاح	وظيفتـــــه
PgDn	الانتقال إلى الصفحة اللاحقة (منسوبة إلى حجم النافذة).
Ctrl-PgUp	الانتقال إلى أول اختيار.
Ctrl-PgDn	الانتقال إلى آخر اختيار.

وكما تلاحظ فإن مفاتيح :

Home – End – Enter – Esc

لاغية وغير مستخدمة.

وعندما يتم استدعاء وظيفة خاصة فإنها تتلقى ثلاث معطيات من الوظيفة ACHOICE( ) هي :

- حالة القائمة (Mode) .
- رقم الاختيار الذي يقف عنده الشريط المضاء.
- رقم السطر الذي يقف عنده الشريط المضاء منسوباً لبداية النافذة.

ويوضح الجدول التالي الحالات الخمس المتصلة بالوظيفة الخاصة ومعنى كل منها:

الحالة (Mode)	معنـاهـــا
0	لم يحصل اختيار.
1	محاولة نقل الشريط المضاء فوق أول اختيار.
2	محاولة نقل الشريط المضاء أسفل آخر اختيار.
3	حدوث خطأ (الضغط على مفتاح خطأ).
4	كل عناصر القائمة لا يمكن اختيارها.

وعندما تقرر استعمال وظيفة خاصة لتتحكم في تنفيذ اختيار القائمة فإن الوظيفة الخاصة ترسل قيما إلى الوظيفة ACHOICE( ) لتوجهها إلى ما يجب تنفيذه. ويوضح الجدول التالي هذه القيم والاجراء الذي ينفذ نتيجة تلقى ACHOICE( ) لها.

القيمة	الاجـــراء المطلوب
0	إلغــــــاء.
1	تنفيذ اختيـــار.
2	التوقف عن التنفيذ.
3	توجيه الشريط المضاء إلى الاختيار التالي الذي يبدأ بآخر حرف أدخل.

الاختلاف عن dBASE III PLUS: لا توجد بها.

مثـــال:

يوضح المثال التالي استخدام الوظيفة ACHOICE( ) في أبسط صورة

M_CHOICE = ACHOICE (03,05,08,12,MAIN)

وفي هذا المثال فإن 03 و05 تحدد الركن اليسار العلوي من القائمة بينما تحدد كلا من 08 و12 الركن اليمين السفلي للقائمة.

أما المثال التالي فيشتمل على برنامج يستخدم الوظيفة ACHOICE لتنفيذ قائمة ذات خمسة اختيارات.

```
* Program: ACH.PRG
CLEAR
*
DECLARE CHARY[5]        && انشاء مصفوفة لتشتمل على اختيارات القائمة
* املأ عناصر المصفوفة بالاختيارات
CHARY[1] = "First  choice"
CHARY[2] = "Second choice"
CHARY[3] = "Third  choice"
CHARY[4] = "Forth  choice"
CHARY[5] = "Fifth  choice"
@ 8,4 TO 14,20 DOUBLE
MYCHOICE = ACHOICE(9,6,14,19,CHARY)  && ضع قيمة الوظيفة فى حقل ذاكرة
                                     && ثم اكتب اختيارات القائمة
DO CASE
   CASE MYCHOICE = 0
        RETURN
   CASE MYCHOICE = 1
        DO PROCA
   CASE MYCHOICE = 2
        DO PROCB
   * <Otheer cases>
ENDCASE
*
PROCEDURE PROCA
@ 24,2 SAY "Good morning"
RETURN
*
PROCEDURE PROCB
@ 24,2 SAY "Good afretnoon"
RETURN
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

MENU TO – ADIR( ) – @...PROMPT

الوظيفة ACOPY()

تنسخ محتويات مصفوفة إلى مصفوفة أخرى.

الشكل العام:

ACOPY (<*array1*>, <*array2*> [,<*expN1*> [,<*expN2*> [,<*expN3*>]]])

حيــث:

<*array1*> : اسم المصفوفة التي سيتم نسخ محتوياتها.

<*array2*> : اسم المصفوفة التي سيتم النسخ إليها.

<*expN1*> : أول عنصر في المصفوفة الأولى مطلوب نسخه.

<*expN2*> : عدد العناصر المطلوب نسخها ابتداء من العنصر المحدد في <*expN1*>

<*expN3*> : أول عنصر في المصيفوفة الثانية.

الشـــرح:

تستخـدم هذه الوظيفة لنسخ محتويات مصفوفة إلى مصفوفة أخرى، ويمكن الاستفادة منها في حالة استخراج بعض عناصر مصفوفة طويلة ونسخها إلى مصفوفة أخرى صغيرة أو في حالة دمج أكثر من مصفوفة في مصفوفة واحدة.

الاختلاف عن *dBASE III PLUS*: لا توجد بها.

مثـــال:

المثال التالي ينسخ العنصر الثاني والثالث والرابع من المصفوفة الأولى (AR1) إلى المصفوفة الثانية (AR2) ابتداءً من العنصر السادس. وبعد التنفيذ ستصير محتويات المصفوفة الثانية AR2 هكذا:

9999911199

```
DECLARE AR1[5], AR2[10]     && انشىء مصفوفتين AR1 و AR2
AFILL(AR1,1)                && املا عناصر المصفوفة الاولى بالرقم 1
AFILL(AR1,9)                && املا عناصر المصفوفة الثانية بالرقم 9
ACOPY(AR1,AR2,2,3,6)        && انسخ ثلاثة عناصر من المصفوفة الاولى
                            && ابتداء من العنصر الثاني الى المصفوفة
                            && الثانية ابتداء من العنصر السادس
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

ADEL( ) – AFILL( )

الوظيفة ADEL()

تحذف أحد عناصر المصفوفة.

الشكل العام:

ADEL (<*array*>,<*expN*>)

حيث:

<*array*> : اسم المصفوفة التي تشتمل على العنصر المراد حذفه.

<*expN*> : رقم العنصر المطلوب حذفه.

الشرح:

عند حذف أحد عناصر مصفوفة فإن باقي عناصر المصفوفة يتم إزاحتها وتقل بمقدار العنصر الذي تم حذفه.

الاختلاف عن *dBASE III PLUS*: لا توجد بها.

مثال:

```
DECLARE ARRVAL[3]        && ARRVAL انشيء مصفوفة ذات ثلاثة عناصر باسم
ARRVAL[1] = 100          && خصص 100 للعنصر الاول
ARRVAL[2] = 150          && خصص 150 للعنصر الثاني
ARRVAL[3] = 200          && خصص 200 للعنصر الثالث
ADEL(ARRVAL[2])          && احذف العنصر الثاني
? ARRVAL[2]              && 200 الاجابة:
? ARRVAL[3]              && هذا الامر سيسبب خطأ لان المصفوفة الآن بها
                         && عنصران فقط
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

ACHOICE( ) – AFILL( ) – LEN( ) – ACOPY( )

الوظيفة *ADIR( )*

تعطى رقماً يوضح عدد الملفات الموجودة على الدليل الحالي والتي تتطابق مع الرموز المعطاة ويمكن بالاضافة إلى ذلك نقل معلومات عن هذه الملفات مثل حجمها وتاريخ ووقت إنشائها إلى مصفوفات أخرى.

الشكل العام:

ADIR (*<sketon>* [*<array1>* [,*<array2>* [,*<array3>* [,*<array4>* [,*<array5>*]]]]])

حيـــث:

<skelton> : رموز تدل على الملفات المطلوبة.

<array1> : اسم المصفوفة التي ستشتمل على أسماء الملفات الموجودة ولذلك فإن عناصرها دائماً حرفية.

<array2> : اسم المصفوفة التي ستشتمل على أحجام الملفات الموجودة ولذلك فإن عناصرها دائماً رقمية.

<array3> : اسم المصفوفة التي ستشتمل على تواريخ إنشاء الملفات الموجودة لذلك فإن عناصرها دائماً رقمية.

<array4> : اسم المصفوفة التي ستشتمل على وقت إنشاء الملفات الموجودة ولذلك فإن عناصرها دائماً حرفية.

<array5> : اسم المصفوفة التي ستشتمل على حالة الملفات (Attribute) ولذلك فإن عناصرها دائماً حرفية.

الشـــرح:

تستخدم هذه الوظيفة لمعرفة عدد الملفات الموجودة على الدليل الحالي، وللحصول على معلومات وافية عن هذه الملفات وهي المعلومات التي تظهر عندما تصدر أمر DIR تحت نظام التشغيل DOS ويستخدم الرمزان الشاملان: * أو ? للدلالة

على أسماء الملفات المطلوبة بنفس المفهوم الذي يستخدمان به مع DOS . يجب وضعهما بين علامتي تنصيص فمثلاً الوظيفة ADIR("*.*") تعطي عدد كل الملفات الموجودة على الدليل الحالي.

بينما تعطي الوظيفة ADIR("ST???.DBF") عدد الملفات التي تنتهي بالاسم الممتد DBF. وتبدأ بحرفي ST متبوعان بثلاثة حروف أخرى.

وعندما تقرر نقل معلومات عن الملفات إلى مصفوفات أخرى (*<Array1>...<array5>*) فيجب أن تعرِّف كل مصفوفة داخل أمر DECLARE أو بأمر مستقل. ويجب أن يكون عدد عناصر المصفوفات الأخرى مطابقاً لعدد الملفات الموجودة.

إذا استخدمت الاختيار *<array5>* ضمن الوظيفة فإن المعلومات التي ستوضع بالمصفوفة عن حالة الملفات هي :

الحالة (ttribute)	معناهـــا
A	الملفات التي يمكن قراءتها وكتابتها (Archive)
D	دليل (Directory)
H	ملفات مخففة (Hidden)
R	ملفات قراءة فقط (Read only)
S	ملفات خاصة بنظام التشغيل (System)

الاختلاف عن dBASE III PLUS : لا توجد بها.

مثـــال:

المثال التالي يستخدم هذه الوظيفة لوضع معلومات عن ملفات قواعد البيانات (DBF.) داخل مصفوفات تمهيداً لعمل نسخ احتياطية (Backup) لها فيما بعد بناء على تاريخ ووقت إنشائها.

```
ضع في حقل ذاكرة عدد ملفات DBF. *
N_DBF = ADIR("*.DBF")
انشيء مصفوفة عدد عناصرها مساو لعدد ملفات DBF. *
DECLARE M_NAMES[N_DBF],M_SIZE[N_DBF],M_DATE[N_DBF],M_TIME[N_DBF]
املأ عناصر المصفوفة بمعلومات الملفات *
ADIR("*.DBF",M_NAMES,M_SIZE,M_DATE,M_TIME])
```

وإليك شرح الأوامر الواردة في هذا المثال:

في الأمر الأول استخدمنا الوظيفة ADIR() نفسها. لتخزين عدد ملفات DBF. في حقل ذاكرة اسمه N.DBF وفي الأمر الثاني أنشأنا ٤ مصفوفات عدد عناصر كل منها مساويا لعدد الملفات الموجودة. وفي الأمر الثالث استخدمنا الوظيفة لنقل أسماء الملفات في المصفوفة الأولى M_NAMES وأحجامها في المصفوفة الثانية M_SIZE وتواريخها في المصفوفة الثانية M_DATE وأوقاتها في المصفوفة الرابعة M_TIME .

وعندما تريد الحصول على معلومات عن مصفوفة واحدة أو مصفوفات معينة، استخدم فراغات محل المصفوفات التي لا تريد معلومات عنها، كما يتضح من المثال التالي:

```
EMPTY = " "
ADIR("*.DBF",EMPTY,EMPTY,M_DATE,EMPTY)
```

في هذه الحالة سيتم نقل تواريخ الملفات فقط إلى المصفوفات. ويمكن تجاهل المصفوفة/ المصفوفات إذا كانت في نهاية الأمر ولا تريد تعبئة محتوياتها فمثلاً الأمر التالي يعطي نفس النتيجة:

ADIR("*.DBF",EMPTY, EMPTY, M_DATE)

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

DECLARE – AFILDS( ) – AFLL( ) – AINS( )

الوظيفة AFIELDS( )

تضع معلومات عن حقول الملف الحالي داخل مصفوفات.

الشكل العام:

AFIELDS (<*array1*> [,<*array2*> [<*array3*> [<*array4*>]]])

حيث:

<*array1*>	:	المصفوفة التي ستشتمل على أسماء الحقول.
<*array2*>	:	المصفوفة التي ستشتمل على أنواع الحقول.
<*array3*>	:	المصفوفة التي ستشتمل على أطوال الحقول.
<*array4*>	:	المصفوفة التي ستشتمل على عدد الأرقام العشرية الموجودة في الحقول الرقمية.

الشرح:

تفيد هذه الوظيفة في تحليل مواصفات ملف قاعدة البيانات وتخزين هذه المواصفات داخل مصفوفات. وهذه الوظيفة تشبه الوظيفة ADIR( ) التي شرحنا معا من قبل إلا أن هذه الوظيفة تضع معلومات عن حقول الملف في المصفوفات في حين تضع الوظيفة ADIR( ) معلومات عن الملفات الموجودة على الدليل. ولذلك فلابد من إنشاء المصفوفات قبل ملئها بمعلومات الحقول (الاسم ـ النوع ـ الطول ـ عدد الأرقام العشرية). ويجب أن تراعي أن يكون عدد عناصر المصفوفة كافيا لتعبئة معلومات حقول الملف. فإذا كان عدد عناصر المصفوفة أقل من عدد حقول الملف فإن الحقول الزائدة لن توضع معلوماتها داخل المصفوفة. وننصح باستخدام الوظيفة FCOUNT( ) لمعرفة عدد حقول الملف وبالتالي تحديد عدد عناصر المصفوفة.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثال:

المثال التالي عبارة عن برنامج قصير يظهر مواصفات الملف المفتوح

```
USE STUDENTS
F_CNT = FCOUNT()      && F_CNT ضع عدد الحقول في حقل:
* انشىء مصفوفة لتضع فيها مواصفات الملف
DECLARE M_NAMES[F_CNT], M_TYPE[F_CNT], M_LEN[F_CNT], M_DEC[F_CNT]
* يمكن استخدام الامر الاتى بدلا من الامرين السابقين
* DECLARE M_NAMES[FCOUNT()],M_TYPE[FCOUNT()], .... etc.
AFIELDS(M_NAMES, M_TYPE, M_LEN,M_LEN, M_DEC)
CNT = 1
* The following statements define a loop do display array contents
* الدوارة التالية لاظهار محتويات المصفوفة
FOR CNT = 1 TO F_CNT
  ? M_NAMES[CNT], M_TYPE[CNT], M_LEN[CNT], M_DEC[CNT]
NEXT
```

وفي هذا المثال استخدمنا الوظيفة( )FCOUNT لتخزين عدد حقول الملف في حقل ذاكرة ثم أنشأنا مصفوفات عدد عناصر كل فيها مساويا لعدد حقول الملف واستخدمنا الوظيفة( )AFIELDS لتخزين مواصفات الحقول داخل المصفوفات وأخيرا أظهرنا محتويات المصفوفات.

إذا أردت الحصول على معلومات عن محتويات مصفوفة واحدة أو مصفوفات معينة استخدم فراغات محل المصفوفات التي لا تريد معلومات عنها هكذا:

```
EMPTY= " "
AFIELDS(M_NAMES,EMPTY,M_LEN)
```

في هذه الحالة ستحصل على أسماء الحقول وأطوالها فقط. وكما تلاحظ تجاهلنا ذكر اسم المصفوفةM-DEC لأنها في آخر المجموعة.

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

ADIR( ) – FCOUNT( ) – DECLARE

الوظيفة AFILL()

تملأ عناصر مصفوفة بقيمة معينة.

الشكل العام:

AFILL (<*array*>,<*exp*> [<*expN1*> [,<*expN2*>]])

حيث:

<*array*>	:	اسم المصفوفة المطلوب تعبئتها.
<*exp*>	:	القيمة التي ستوضع في المصفوفة.
<*expN1*>	:	ترتيب العنصر داخل المصفوفة الذي ستبدأ عنده التعبئة.
<*expN2*>	:	عدد العناصر التي سيتم تعبئتها.

الشرح:

باستخدام هذه الوظيفة يمكن تعبئة عنصر أو أكثر داخل المصفوفة بقيمة معينة فإذا لم تحدد العنصر أو العناصر فسيتم تعبئة كل عناصر المصفوفة بنفس القيمة. وتستخدم هذه الوظيفة في الغالب لوضع قيم تلقائية داخل عناصر المصفوفة.

الاختلاف عن *dBASE III PLUS*: لا توجد بها.

مثال:

المثال التالي يستخدم الوظيفة AFILL( ) لملء كل أو بعض عناصر المصفوفة.

```
DECLARE FIX[10]            && FIX انشىء مصفوفة ذات عشرة عناصر باسم:
AFILL(FIX,"Balance: ")     && Balance: املأ كل عناصر المصفوفة بكلمة
AFILL(FIX,.07,1,3)         && .07 املأ العناصر الثلاثة الاولى بالرقم
AFILL(FIX,.10,4,3)         && .10 املأ العناصر 3 و 4 و 5 بالرقم
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

DECLARE – STORE – ADIR( )

الوظيفة AINS()

تضع عنصر بين عناصر مصفوفة.

الشكل العام:

AINS (<*array*> ,<*expN*>)

حيــــث:

<*array*> : اسم المصفوفة.

<*expN*> : المكان الذي سيوضع فيه العنصر الجديد داخل المصفوفة.

الشـــــرح:

تستخدم هذه الوظيفة لوضع عنصر معين بين عناصر مصفوفة موجودة والعنصر الجديد لا تخصص له أية قيمة. ويتسبب في ازاحة باقي عناصر المصفوفة ولذلك يجب أن تكـون المصفـوفـة كافية لاحتوائه وإلا فإن العنصر الأخير سيضيع. ويجب تعبئة العنصر الجديد بأية قيمة قبل استخدامه لأن استخدام عنصر لا يشتمل على قيمة معينة يسبب خطأ.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثــــال:

لاحظ في المثال التالي بعد إضافة عنصر جديد محل العنصر رقم ٢ أزيح العنصر رقم ٢ مكان العنصر رقم ٣.

```
DECLARE EXPAND[4]          && EXPAND انشيء مصفوفة باسم
EXPAND[1] = "Magdi"        && Mgdi خصص للعنصر الاول كلمة
EXPAND[2] = "Ali"          && Ali خصص للعنصر الثاني كلمة
EXPAND[3] = "Maher"        && Maher خصص للعنصر الثالث كلمة
? EXPAND[2]                && Ali   الاجابة
AINS(EXPAND, 2)            && اضم عنصرا جديدا قبل العنصر الثاني
? EXPAND[3]                && Ali   الاجابة
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

DECLARE - ADEL( ) - LEN( )

الوظيفة ALIAS()

تحدد اسم أو رمز بديل لملف قاعدة البيانات المفتوح.

الشكل العام:

ALIAS <*expN*>)

حيـــث:

<*expN*> : رقم المنطقة (Workarea) التي تريد السؤال عنها.

الشـــرح:

عندما تفتح ملف قاعدة البيانات يجوز أن تخصص اسما آخرا ليكون بديلاً لاسم ملف قاعدة البيانات وهو المقصود بكلمة Alias . فإذا لم تحدد اسم بديل لملف قاعدة البيانات عند فتحه بأمر USE فإن قاعدة البيانات تخصص نفس الاسم الملف للوظيفة ALIAS( ) .

إذا لم تحدد رقم المنطقة مع الوظيفة فستفهم قاعدة البيانات أنك تشأل عن الملف الموجود في المنطقة الحالية. وإذا لم تجد قاعدة البيانات ملفاً مفتوحاً فستعطيك فراغ.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

في المثـال التـالي عنـد السؤال عن الملف الموجود بالمنطقة رقم ١ تجيب قاعدة البيانات بالاسم البديل المخصص مع أمر USE . وعند السؤال عن الاسم البديل (Alias) بدون تحديد رقم المنطقة تجيب قاعدة البيانات باسم الملف الموجود في المنطقة الحالية.

```
USE INVENT ALIAS MASTER
SELECT 2
USE SALE
SELECT 3
USE PURCH
?ALIAS(1), ALIAS(2), ALIAS()    && MASTER SALE PURCH  الإجابة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

USE – SELECT – SELECT( )

الوظيفة ALLTRIM()

تحذف الفراغات الموجودة على يمين ويسار عبارة حرفية .

الشكل العام:

ALLTRIM (<*expC*>)

حيـــث :

<*expC*> : العبارة المقصودة .

الشـــرح:

هذه الوظيفة بديل للوظيفة :

LTRIM (TRIM(<*exp*>))

الموجودة في dBASE III PLUS

الاختلاف عن *dBASE III PLUS* : لا توجد بها .

مثـــال :

```
OK = SPACE(10)+"Come here"+SPACE(10)
? OK+"*"              &&          Come here          *   الإجابة
? ALLTRIM(OK)+"*"     && Come here*   الإجابة
```

المكتبة : EXTEND.LIB

الأوامر ذات الصلة:

LTRIM( ) – TRIM( ) – ?

الوظيفة ALTD()

تسمح باستدعاء مكتشف الأخطاء (Debugger) أو تلغي إمكانية استدعائه.

الشكل العام:

ALTD ([<*expN*>])

حيث:

<*expN*> : رقم يوجه إلى مكتشف الأخطاء لاستدعائه أو لتعطيل استدعائه.

الشرح:

تسمح هذه الوظيفة بتوجيه مكتشف الأخطاء لاتخاذ قرار معين بناء على الرقم الذي يدخل معها. وإليك الأرقام التي يمكن إدخالها محل <*expN*> ومعنى كل منها.

الرقم	معنـــاه
0	تعطيل استدعاء مكتشف الأخطاء داخل البرنامج.
1	تمكين استدعاء مكتشف الأخطاء داخل البرنامج.
2	استدعاء مكتشف الأخطاء واستدعاء حقول الذاكرة الخاصة معه.

فإذا استخدمت الوظيفة بدون معطيات هكذا: ALTD( ) فإنها تستدعي مكتشف الأخطاء بعد حفظ الشاشة الموجودة.

ويشترط لاستخدام هذه الوظيفة داخل البرنامج أن تربط برنامج DEBUG.OBJ الذي يأتي ضمن حزمة «كلبر» ضمن برنامجك. وهذه الوظيفة بديل لضغط مفتاح ALT-D لاستدعاء مكتشف الأخطاء. فمثلاً: ALTD( ) بديل لضغط

مفتـاح Alt-D أثنـاء تنفيذ الـبرنامج ، و ALTD(0) تجعل ضغط مفتاح Alt-D عديم الجدوى أثناء تنفيذ البرنامج .

الاختلاف عن dBASE III PLUS : لا توجد بها .

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SETCANCEL( )

الوظيفة ASC()

تعطى الشفرة (ASCII) المقابلة للحرف الأول الموجود في عبارة.

الشكل العام:

ASC (<*expC*>)

حيث:

<*expC*> : حرف أو عبارة.

الشرح:

تعطى هذه الوظيفة رقما يمثل الشفرة الأمريكية (ASCII Code) المقابلة للحرف الأول أو الرمز الموجود في العبارة <*expC*>

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثال:

يوضح المثال التالي كيف تعرف الشفرة المقابلة لحرف معين أو للحرف الأول من كلمة.

```
? ASC("Magdi")          && 77   الاجابة
? ASC("M")              && 77   الاجابة
? ASC("m")              && 109  الاجابة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

CHR() – INKEY()

الوظيفة ASCAN()

تبحث عن أول عنصر داخل مصفوفة يتطابق مع عبارة معينة.

الشكل العام:

ASCAN (<*array*>, <*exp*> [<*expN1*> [<*expN2*>]])

حيـــث:

<*array*>	:	اسم المصفوفة.
<*exp*>	:	القيمة التي سيتم البحث عنها.
<*expN1*>	:	رقم العنصر الذي سيبدأ البحث عنده.
<*expN2*>	:	عدد العناصر التي سيتم البحث فيها.

الشـــرح:

تبحث هذه الوظيفة داخل كل عناصر المصفوفة عن العنصر المقابل للقيمة الداخلة. فإذا استخدم <*expN1*> و/أو <*expN2*> معها فسيتم البحث ابتداء من رقم العنصر بعدد العناصر المحددة وهي تشبه أمر SEEK أو FIND الذي يبحث داخل الملف عن عبارة معينة ولذلك فإذا كان أمر SET EXACT في وضع ON فيجب تطابق عبارة البحث والعنصر الذي تبحث عنه تماماً. وعندما تجد قاعدة البيانات العنصر الذي تبحث عنه داخل المصفوفة فإنها تعطيك رقمه أما إذا لم تجده فتعطيك الرقم صفر.

الاختلاف عن *dBASE III PLUS*: لا توجد بها.

مثـــال:

المثال التالي ينشىء مصفوفة تتكون ٢٠ عنصر تشتمل على أرقام حسابات العملاء ثم يبحث عن رقم عميل داخل المصفوفة.

```
DECLARE CUST[20]
* <Commands to load data into the array>
? ASCAN(CUST,"DB-123")
```

فبفرض أن ترتيب الحساب BB-123 هو العاشر داخل المصفوفة فستحصل على الرقم 10 .

ولتبحث داخل المصفوفة في العناصر من ١٠ ـ ١٥ فقط أدخل الأمر هكذا:

```
? ASCAN(CUST,"DB-123",10,15)
```

فإذا كان الحساب غير موجود بالمصفوفة فستحصل على الرقم صفر.

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

FIND – SEEK – SET EXACT – AFILL( )

الوظيفة ASORT()

ترتب عناصر مصفوفة ترتيبا تصاعديا.

الشكل العام:

ASORT (<*array*> [,<*expN1*> [,<*expN2*>]])

حيث:

<*array*> : اسم المصفوفة المطلوب ترتيب عناصرها.
<*expN1*> : رقم العنصر الذي سيبدأ الترتيب عنده.
<*expN2*> : عدد العناصر التي سيتم ترتيبها.

الشرح:

تستخدم هذه الوظيفة لترتيب عناصر مصفوفة ترتيبا تصاعديا فإذا أضيف إليها <*expN1*> فسيبدأ الترتيب من رقم هذا العنصر، وإذا أضيف إليها <*expN2*> فسينتهي الترتيب عند رقم هذا العنصر ويجب أن تكون عناصر المصفوفة التي سيتم ترتيبها من نفس النوع.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثال:

المثال التالي يظهر الملفات مرتبة ترتيباً أبجدياً ولذلك فهو يستخدم أولاً الوظيفة ADIR() لتحديد عدد عناصر المصفوفة ثم يضع أسماء الملفات داخل المصفوفة. وأخيراً يرتب أسماء الملفات داخل المصفوفة.

```
DECLARE SORTED[ADIR("*.*")]   && SORTED انشىء مصفوفة باسم
                              && عدد عناصرها يساوى عدد الملفات
                              && الموجودة على الدليل الحالى
ADIR("*.*",SORTED)            && املأ عناصر المصفوفة بأسماء الملفات
ASORT(SORTED)                 && رتب عناصر المصفوفة
```

فإذا أردت ترتيب العناصر من ١٠ إلى ٢٥ فقط استخدم الوظيفة بهذا الشكل

```
ASORT(SORTED,10,16)
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

DECLARE – ADIR( ) – AFILL( )

الوظيفة AT()

تحدد أين تبدأ عبارة أو كلمة داخل عبارة أخرى.

الشكل العام:

AT(<*expC1*>, <*expC2*>)

حيـــث:

<*expC1*> : الكلمة المطلوب البحث عنها.

<*expC2*> : العبارة المطلوب البحث فيها.

الشــــرح:

تعطي هذه الوظيفة رقماً يوضح مكان بداية عبارة داخلية <*expC1*> داخل عبارة رئيسية أخرى <*expC2*> . فإذا لم تكن العبارة الداخلية موجودة ضمن العبارة الرئيسية التي تبحث فيها فستعطيك الرقم 0 .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

المثال الآتي يوضح أين تبدأ كلمة

```
STORE "This is a cat" TO STRING
STORE "his" TO CAT
? AT(CAT,STRING)              && 2  الاجابة
? AT("dog","This is a cat")   && 0  الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

LEFT( ) – RIGHT( ) – SUBTR( )

الوظيفة BOF()

توضح هل المؤشر عند بداية الملف أم لا.

الشكل العام:

BOF()

الشـــرح:

تستخدم هذه الوظيفة في التطبيقات أو البرامج التي تقرأ ملف قاعدة البيانات بالعكس أي من النهاية إلى البداية. وهي تعطي دلالة عما إذا كان المؤشر وصل إلى بداية الملف أم لا. فإذا كان المؤشر وصل إلى علامة البداية ـ وهذه العلامة تسبق السجل الأول في الملف ـ فستعطي هذه الوظيفة القيمة المنطقية .T. بمعنى صح (True) وإلا فستعطي القيمة .F. بمعنى خطأ (False) .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

يتضح من المثال الآتي أن علامة البداية في الملف تسبق السجل الأول بمكان لسجل واحد.

```
USE STOCK
? RECNO()                && 1  الإجابة
? BOF()                  && .F.  الإجابة
SKIP - 1
? BOF()                  && .T.  الإجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

EOF() – REC NO() – SKIP

الوظيفة BROWSE( )

تشبه أمر BROWSE الموجود في dBASE III PLUS

الشكل العام:

BROWSE ([<expN1> ,<expN2> ,<expN3> ,<expN4>])

حيـــث:

<expN1> : رقم السطر الذي سيبدأ عنده مستطيل الاظهار.

<expN2> : رقم العمود الذي سيبدأ عنده مستطيل الاظهار.

<expN3> : رقم السطر الذي سينتهي عنده مستطيل الاظهار.

<expN4> : رقم العمود الذي سينتهي عنده مستطيل الاظهار.

الشــــرح:

تستخدم هذه الوظيفة بديلاً لأمر BROWSE لاظهار بيانات السجل داخل مستطيل يتم تحديد مساحته. ولن تجد هذه الوظيفة ضمن الوظائف المدرجة في كتاب الشركة المنتجة لوجود وظيفة أخرى أكثر كفاءة وأكثر إمكانيات هي الوظيفة DBEDIT( )

الاختلاف عن *dBASE III PLUS* : غير موجودة بها وأمر BROWSE غير موجود في Clipper

مثـــال:

المثال التالي يظهر سجل داخل مستطيل يبدأ من السطر الأول عمود ٥ وينتهي عند السطر ٢٣ والعمود ٧٥.

```
USE INVENT
BROWSE(1,5,23,70)
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

DBEDIT( )

الوظيفة CDOW()

تعطى اسم اليوم داخل الأسبوع المسجل بالتاريخ.

الشكل العام:

CDOW (<*expD*>)

حيــــث:

<*expD*> : حقل تاريخي أو عبارة تاريخية.

الشـــــرح:

تستخدم هذه الوظيفة لمعرفة اسم اليوم ضمن أيام الأسبوع حسب التاريخ المسجل بالحاسب أو حسب التاريخ المسجل في حقل تاريخي ولذلك فإن بيانات <*expD*> لابد أن تكون تاريخية.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

لمعرفة اسم اليوم من خلال التاريخ المسجل بالحاسب:

```
? CDOW(DATE())          && Friday  الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

DOW() – DAY()

الوظيفة CHR( )

تعطي الحرف أو الرمز المقابل للشفرة الأمريكية (ASCII)

الشكل العام:

CHR(<*expN*>)

حيـــث:

<*expN*> : رقم أو تعبير رقمي يقع بين ١ و ٢٥٤.

الشـــرح:

تعمل هذه الوظيفة عكس وظيفة ASC( ) فهي تظهر الحرف أو الرمز المقابل للشفـرة الأمـريكية (ASCII Code) الموجود بالأمر. ولذلك فهي تتيح لك الفرصة لاظهار الحروف التي لا تستطيع إدخالها من لوحة المفاتيح مثل علامات الأسهم أو صوت الجـرس. أو العـلامـات التي تستخدم في الرسومات والتي تظهر فقط على الطابعات أو الشاشات.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

لكي تسمع صوت الجرس متبوعا بعبارة Take Care :

```
? CHR(7) + "Take Care"
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

ASC( ) – KEYBOARD

الوظيفة CMONTH( )

لاظهار اسم الشهر المسجل بالتاريخ .

الشكل العام:

CMONTH (<*expD*>)

حيـــث:

<*expD*> : حقل تاريخي أو عبارة تاريخية .

الشـــرح:

تظهر اسم الشهر المسجل ضمن حقل تاريخي أو اسم الشهر الموجود بالتاريخ المسجل بالحاسب أو اسم الشهر الموجود في حقل ذاكرة تاريخي .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــال:

لمعرفة اسم الشهر الموجود بحقل DATE في ملف STOCK.dbf في السجل الثالث:

```
USE STOCK
GO 3
? CMONTH(DATE)                && April  الاجابة
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة:

MONTH( )

الوظيفة COL()

تعطي رقم العمود الذي يقف عنده المؤشر على الشاشة.

الشكل العام:

COL( )

الشـــرح:

تستخدم هذه الوظيفة غالبا داخل البرامج لتوجيه مؤشر الشاشة إلى عمود معين على الشاشة ـ ومعروف أن الشاشة تقسم إلى ٨٠ عمودا من صفر إلى ٧٩. تستخدم هذه الوظيفة في الغالب لتحديد مكان معين (address) على الشاشة فمثلا إذا كان المؤشر عند العمود العاشر فإن الأمر 1,COL( )+10 @سيوجه المؤشر إلى السطر الأول والعمود العشرين.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

أدخل الأوامر التالية داخل البرنامج لكي تظهر العبارة "Enter your selection" عند السطر الخامس والعمود الخامس وبعد خمسة فراغات أخرى على نفس السطر تظهر العبارة "Y or N"

```
@ 5,5       SAY "Enter your selection"
@ 5,COL + 5 SAY "Y or N"
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@ – PCOL( ) – ROW( ) – PROW( )

الوظيفة CTOD()

تحول عبارة حرفية في شكل تاريخ إلى تاريخ.

الشكل العام:

CTOD(<*expC*>)

حيث:

<*expC*> : بيانات حرفية تأخذ شكل تاريخ.

الشــرح:

تحول هذه الوظيفة بيانات أدخلت إلى الحاسب كبيانات حرفية ولكنها في شكل تاريخ إلى بيانات تاريخية. وتأخذ البيانات الحرفية المطلوب تحويلها إلى بيانات تاريخية الشكل الآتي:

mm/dd/yy

ويجب الانتبـاه إلى أن هذا الشكل يتغير حسب تأثير أمر SET DATE وأمر SET CENTURY وتفترض قاعدة البيانات أن قرن التاريخ هو القرن العشرين ما لم تغيره بأمر SET CENTURY .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـال:

يبين المثال التالي تأثير هذه الوظيفة على نوع البيانات.

```
STORE "10/10/90" TO "THISDATE"
? TYPE(THISDATE)                  && C  الاجابة
STORE CTOD(THISDATE) TO MYDATE
? TYPE(MYDATE)                    && D  الاجابة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

DTOC( ) – SET CENTURY – SET DATE – DTOS( )

الوظيفة CURDIR()

لاظهار أو لمعرفة اسم الدليل الحالي.

الشكل العام:

CURDIR([<*expC*>])

حيـــث:

<*expC*> : اسم مشغل القرص الذي يشتمل على الدليل.

الشـــرح:

تظهر هذه الوظيفة اسم الدليل الموجود على مشغل القرص المختار فمثلاً: ?CURDIR("C") تظهر اسم الدليل الموجود على مشغل القرص C هكذا مثلاً: CLIPPER \APP ، وإذا لم تشتمل الوظيفة على اسم مشغل القرص فإنها تظهر اسم الدليل الموجود على القرص المخصص للعمل. أما إذا كنت تعمل تحت الدليل الرئيسي أو إذا كان اسم مشغل قرص خطأ فلن تحصل على شي.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال الآتي جزء من برنامج يختص بتركيب النظام تحت دليل معين ولذلك فهو يتأكد من وجود هذا الدليل قبل عملية التركيب

```
IF .NOT. CURDIR() = "CLIPPER\APP"
  ?
  ?"Please make directory named: \CLIPPER\APP ON DRIVE C"
  QUIT
ENDIF
```

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

SET DEFAULT – SET PATH

الوظيفة DATE()

تظهر التاريخ المسجل بالحاسب.

الشكل العام:

DATE( )

الشـــرح:

تظهر التاريخ المسجل بالحاسب حسب الشكل المختار بأمر SET DATE .

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

لاظهار تاريخ اليوم كما هو مسجل بالحاسب ولكن بالشكل العربي:

```
SET DATE ANSI
? DATE()                  && 90.09.13  الاجابة
? DATE() + 10             && 90.09.23  الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET CENTURY – SET DATE

الوظيفة DAY()

تظهر ترتيب اليوم في الشهر كما هو مسجل بالتاريخ.

الشكل العام:

DAY(<*expD*>)

حيــــث:

<*expD*> : حقل أو عبارة تاريخية.

الشــــرح:

يظهر رقم يدل على ترتيب اليوم داخل الشهر من حقل أو عبارة تاريخية سواء كانت حقل بالملف أو بالذاكرة أو التاريخ المسجل بالحاسب.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثــــال:

```
? DATE()            && 09/13/90  الإجابة
? DAY(DATE())       && 13  الإجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

CDOW( ) – DOW( )

الوظيفة DBEIT( )

تسمح بإظهارر وتعديل السجل بطريقة مشابهة لأمر BROWSE الموجود في «دي بيس ثري بلاس».

الشكل العام:

DBEDIT([<*expN1*> [,<*expN2*> [<*expN3*> [<*expN4*>]]]]
[<*array1*>] [<*expC*>] [<*array2*>/<*expC*>] [<*array3*>/<*expC*>]
[<*array4*>/<*expC*>] [<*array5*>/<*expC*>]
[<*array6*>/<*expC*>] [<*array7*>/<*expC*>]

وسنتناول في الشرح الآتي شرح المفردات الواردة بالوظيفة.

الشـــرح:

تستخـدم هذه الوظيفة للسماح للمستخدم بإظهار وتعديل حقول السجل من خلال نافذة تظهر على الشاشة مثل تلك التي تظهر نتيجة لأمر BROWSE في dBASE III PLUS إلا أنها هنا تعطي المستخدم صلاحيات أكثر في التحكم في حجم النافذة التي تظهر على الشاشة والحقـول التي تظهـر داخلها وشكل إظهارها وصلاحيات تعديل محتوياتها والأسماء المختارة للحقـول التي ستـظهر وفي الخطوط المستخدمة كفاصل بين عناوين الحقول وبياناتها أو بين الحقول نفسها. . . الخ. مما سيتضح من خلال الشرح التالي للمفردات الواردة بالوظيفة.

<expN4>...<*expN1*> : تحدد حجم النافذة التي ستظهر على الشاشة ـ مشتملة على الحقول ـ ومكان ظهورها حيث: <*expN1*> رقم أعلى سطر تبدأ عنده النافذة على الشاشة، <*expN2*> رقم العمود الأيسر أي يحددان الركن اليسار العلوي من النافذة، أما <*expN3*> فهـو رقم آخر سطر تنتهي عنده النافذة، <*expN4*> رقم العمود الأيمن أي يحددان الركن اليمين السفلي من النافذة. إذا لم تحدد مكان النافذة فستحتل كل الشاشة.

<*expC*> : اسم وظيفة خاصة (User Defined Function) تشتمل على مجموعة الأوامر التي تتحكم في توجيه المؤشر واستخدام المفاتيح ويجب أن توضع بين علامتي تنصيص هكذا "" . وهذا الاختيار ضروري عندما ترغب في تعديل محتويات الحقول. فإذا لم تحدد اسم الوظيفة الخاصة وتعدها فإن مجرد ضغط مفتاح الادخال سينهي مفعول الوظيفة DBEDIT( ) .

ويجب إعداد الوظيفة الخاصة لتتعرف على آخر حرف تم استخدامه من لوحة المفاتيح من خلال الـوظيفـة LASTKEY( ) وحالة الوظيفة DBEIT( ) أو (Mode) وتفحص الوظيفة الخاصة الحالة (Mode) والقيمة الموجودة بالوظيفة LASTKEY()ثم ترسل قيمة إلى DBEDIT( ) (توضح ما يجب أن يتم.

ويوضح الجدول التالي الحالات (Modes) التي ترسلها DBEDIT( ) إلى الوظيفة الخاصة.

الحالة Mode	معناهـــا
0	تم استخدام كل المفاتيح ولا توجد مفاتيح منتظرة الضغط عليها.
1	محاولة المستخدم تحريك المؤشر قبل بداية الملف.
2	محاولة المستخدم تحريك المؤشر بعد نهاية الملف.
3	الملف لا يشتمل على سجلات.
4	خطأ أثناء استخدام أحد المفاتيح.

بينما يوضح الجدول التالي القيم التي ترسلها الوظيفة الخاصة إلى الوظيفة DBEDIT( ) ومعنى كل منها.

القيمة	معناهـــــا
0	انهاء عمل الوظيفة DBEDIT( )
1	استمرار عمل الوظيفة DBEDIT( )
2	إعادة رسم الشاشة والاستمرار في العمل.

<array2>/<expC> : اسم المصفوفة التي تشتمل على أشكال إظهار بيانات الحقول (PIC-TURE) وصور تعديلها. إذا استخدمت *<expC>* فإن جميع الحقول ستظهر بنفس الشكل (PICTURE) راجع أمر GET...SAY...@ لتتعرف على صور إظهار وتعديل الحقول.

<array3>/<expC> : اسم مصفوفة تشتمل على أسماء للحقول التي ستظهر، إذا استخدم *<expC>* فإن الاسم المختار سيظهر فوق جميع الحقول.

<array4>/<expC> : اسم مصفـوفـة تشتمـل على الحروف التي ستستخدم كفاصل بين العنـاوين والأعمدة التي تحتها. إذا استخدمت *<expC>* فإن الحرف المشار إليه سيستخدم تحت جميع العناوين.

<array5>/<expC> : اسم مصفـوفـة تشتمـل على الحروف التي ستستخدم كفاصل بين الأعمدة. إذا استخدمت *<expC>* فإن الحرف المشار إليه سيستخدم بين كل الأعمدة.

<array6>/<expC> : اسم مصفوفة تشتمل على الحروف التي ستستخدم لرسم خط بين الحقول والتذييل (Footer) المختار للشاكة. إذا استخدمت *<expC>* فإن الحرف المشار إليه سيستخدم لرسم كل الخطوط بين الحقول والتذييلات.

<array7>/<expC> : اسم مصفـوفـة تشتمـل على عبـارات تظهر أسفل الحقول كتذييل للحقول (Column Footer) . إذا استخدمت *<expC>* فإن نفس العبارة ستظهر تحت كل الحقول.

ويجب الانتباه إلى أن جميع هذه المفردات اختيارية فيجوز اختيار واحد منها أو أكثر حسب حاجتك، فإذا لم تخترها كلها أو لم تختر أحدها فستخصص قاعدة البيانات قيما تلقائية للاختيار/ الاختيارات غير الموجودة.

والمثال التالي يوضح أبسط صور استخدام هذه الوظيفة إذا أردت أن تخصص قاعدة البيانات قيما تلقائية للاختيارات كلها

USE STUDENTS

DBEIT( )

ويمكن استخدام الوظيفة ( )DBEDIT داخل أخرى إذا أردت إظهار أكثر من نافذة في نفس الوقت.

الاختلاف عن dBASE III PLUS: لا توجد بها.

مثـــال:

المثـال التـالي عبـارة عن برنامج يستخدم لنفس الغرض الذي يمكن الحصول عليه باستخدام أمر BROWSE الموجود في dBASE III PLUS ويشتمل في داخله على وظيفة خاصة.

```
* Program: EDIT.PRG
ex3a.txt 11/12/90

* Using DBEDIT() function insted of BROWSE command in dBASE III PLUS
USE STUDENTS
SELEC 2                      && 2 اختر المنطقة رقم
USE COURSES INDEX I_NUM
SELECT 1                     && 1 اختر المنطقة رقم
* Relate the current file with COURSES according to STUDENTNO
SET RELATION TO STUDENTNO INTO COURSES
```

```
* Define array named ARFIELD to hold field names and other named
* ARPICT to hold pictures and functions
DECLARE ARFIELD[3],ARPICT[3]
ARFIELD[1] = "NAME"             && املأ العنصر الاول من المصفوفة
ARFIELD[2] = "ADDRESS"          && املأ العنصر الثانى من المصفوفة
ARFIELD[3] = "B->COST"          && املأ العنصر الثالث من المصفوفة
                                && باسم حقل موجود فى المنطقة B
ARPICT[1] = "@!"                && اظهر محتويات العنصر الاول بالحروف
                                && الكبيرة
ARPICT[2] = "@A"                && اقبل حروف فقط فى العنصر الثانى
ARPICT[3] = "999,999.99"        && لادخال فاصلة بين كل ثلاثة ارقام
DBEDIT(2,5,23,75,ARFIELD,"ED_FUN",ARPICT)
CLEAR
CLEAR ALL
RETURN
*
* الوظيفة الخاصة التالية تستدعى من داخل البرنامج
FUNCTION ED_FUN
PARAMETERS MODE,CUR_FLD
PRIVATE CURRENT
CURRENT = ARFIELD[CUR_FLD]     && ضع الحقل الموجود تحت المؤشر فى الذاكرة
                               && لغرض تعديله
DO CASE                        && نفذ احدى الحالات الآتية
  CASE MODE = 0             && لو الحالة القادمة من DBEDIT() تساوى صفرا
    @ 24,2 SAY "Don't edit this record "   && اظهر رسالة
    RETURN(1)                  && استمر فى التنفيذ
  CASE MODE = 1 .OR. MODE = 2
    ? REPLICATE(CHR(7),2)      && رن الجرس مرتين
    @ 24,2 SAY "Begining or End of file"   && اظهر رسالة
    RETURN(1)                  && استمر فى التنفيذ
  CASE MODE = 3
    ? REPLICATE(CHR(7),2)      && رن الجرس مرتين
    @ 24,2 SAY "Empty file           "   && اظهر رسالة
    RETURN(0)                  && انه البرنامج
  CASE LASTKEY() = 27         && اذا ضغط مفتاح Esc
    RETURN(0)                 && انه البرنامج
  CASE LASTKEY() = 13         && اذا ضغط مفتاح Enter
    * Edit the record
    M_ROW = ROW()
    M_COL = COL()
    @ 24,2 SAY "Editing record number    " + LTRIM(STR(RECNO()))
```

```
    @ M_ROW,M_COL GET &CURRENT  && عدل السجل الحالى
    READ                        && ضع المؤشر عند اول GET
    KEYBOARD CHR(4)             && حرك المؤشر حرفا
    RETURN(1)                   && استمر فى التنفيذ
  OTHERWISE                     && استمر فى التنفيذ اذا استخدم
    RETURN(1)                   && اى مفتاح آخر غير المذكورة هنا
ENDCASE
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

MEMOEDIT( ) – ADIR( ) – AFIELDS( )

الوظيفة DBFILETER( )

تعطي العبارة المستخدمة مع أمر SET FILTER TO

الشكل العام:

DBFILTER( )

الشـــرح:

عنـدما تصدر أمر SET FILTER TO <*Condition*> فإن العبارة التي تحدد الحالة المطلوبة تخزن في الذاكرة ويتعامل البرنامج مع الملف بحالته الجديدة، فإذا أردت أن تعرف العبارة التي تتحكم في بيانات الملف <*Condition*> استخدم هذه الوظيفة. فإذا لم تكن هناك حالة مختارة للتحكم في بيانات الملف فلن تحصل على شيء.

ويمكن استخدام هذه الوظيفة مع وظيفة *DBRELATION( )* ووظيفة *INDEXORD( )* بديلاً لأمر DISPLAY STATUS الموجود في «دي بيس ثري بلاس» للحصول على بعض المعلومات التي يظهرها.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثال:

في هذا المثال يشتمل ملف STAT.MEM على الحالة التي تتحكم في بيانات الملف (Filter) واسم الملف المرتبط مع الملف الحالي (Relation) واسم ملف الفهرس الرئيسي (Index) .

```
STORE DBFILTER()   TO M_FIL
STORE DBRELATION() TO M_REL
STORE INDEXORD()   TO M_IND
SALE ALL LIKE M_*  TO STAT.MEM
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET FILTER – DBRELATION( ) – DBRSELECT

الوظيفة DBRELATION( )

تعطي اسم الحقل المتخذ أساساً لربط ملفات قواعد البيانات باستخدام أمر SET RELATION .

الشكل العام:

SET RELATION(<*expN*>)

حيـــث:

<*expN*> : رقم يوضح ترتيب العلامة الموجودة في أمر SET RELATION

الشـــرح:

استخدم هذه الوظيفة إذا كنت تريد معرفة الحقل الذي يربط ملفي قاعدة البيانات عنـد ربط أكثر من ملف بأمر SET RELATION . فإذا لم تكن هناك علاقة بين الملفات فلن تحصل على شيء.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثال:

المثـال التـالي يسأل عن العلاقة الموجودة في المنطقة رقم ٢ وعن الموجودة في المنطقة رقم ٤ التي لا يوجد بها شيء.

```
USE INVENT INDEX ITEM_NO
SELECT 2
USE SALE INDEX I_SALE
SELECT 3
USE PURCH INDEX I_PUR
SELECT 1
SET RELATION TO ITEM_NO INTO SALE, TO ITEM_NO INTO PURCH
? DBRELATION(2)            && ITEM_NO  الإجابة
? DBRELATION(4)            && لن تحصل على شيء
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

SET RELATION – DBFILTER( ) – DBRSELECT

الوظيفة DBRSELECT( )

تعطي رقم المنطقة التي يوجد بها الملف المرتبط مع الملف الحالي بامر SET RELATION .

الشكل العام:

DBRSELECT (<*expN*>)

حيـــث:

<*expN*> : موقع الملف المطلوب داخل أمر SET RELATION

الشـــرح:

تعطي هذه الوظيفة رقما يوضح رقم المنطقة التي يوجد بها الملف المرتبط مع الملف الحالي بأمر SET RELATION . ويمكن استخدام هذه الوظيفة مع وظيفة DBRELATION( ) ووظيفة DBFILTER لتخزين معلومات داخل ملف تفيد عن حالة الملفات وعلاقاتها معاً كبديل للمعلومات التي تظهر بأمر DISPLAY STATUS في «دي بيس».

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثال:

```
USE INVENT INDEX ITEM_NO
SELECT 2
USE SALE INDEX I_SALE
SELECT 3
USE PURCH INDEX I_PUR
SELECT 1
SET RELATION TO ITEM_NO INTO SALE, TO ITEM_NO INTO PURCH
? DBRELATION(2)                 &&  ITEM_NO  الاجابة
? DBRSELECT(2)                  &&  3   الاجابة
```

لأن الملف الثاني في أمر *SET RELATION* وهو *PURCH.DBF* موجود في المنطقة رقم ٣.

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET RELATION – DBFILTER( ) – DBRELATION( )

الوظيفة DELETED( )

تعطي دلالة عن السجلات المعلمة لأغراض الحذف.

الشكل العام:

DELETED( )

الشــــرح:

تعطي هذه الوظيفة القيمة المنطقية .T. بمعنى صح (True) إذا كان السجل الذي يقف عنده المؤشر داخل الملف معلما لأغراض الحذف بأمر DELETE . وإلا فتعطي .F. وتفيد هذه الوظيفة إذا استخدمت داخل البرنامج لاظهار كل السجلات المعلمة لأغراض الحذف أو لاستبعادها أو لاجراء عمليات حسابية عليها أو لمعرفة عددها . . . الخ .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــــال:

المثال التـالي يستخدم هذه الوظيفة لمعرفة هل السجل الحالي معلم بغرض الحذف أم لا . ثم يستخدم لمعرفة عدد السجلات المعلمة لأغراض الحذف.

```
USE STOCK
DELETE
? DELETED()              && .T.  الاجابة
COUNT FOR DELETED()      && 1   الاجابة
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

DELETE – SET DELETE

الوظيفة DESCEND( )

تسمح بإنشاء فهرس مرتب ترتيبا تنازليا.

الشكل العام:

DESCEND(<*exp*>)

حيث:

<*exp*> : الحقل المتخذ أساسا لترتيب السجلات.

الشرح:

تسمح هذه الوظيفة بترتيب السجلات داخل الملف ترتيبا تنازليا طبقا للعبارة أو الحقل المتخذ أساسا لترتيب البيانات (Key index) ويجوز ترتيب البيانات الحرفية أو الرقمية أو التاريخية باستخدام هذه الوظيفة.

الاختلاف عن dBASE III PLUS : لا توجد بها.

مثال:

المثال التالي ينشىء ملف فهرس مرتبا ترتيبا تنازليا طبقا لبيانات حقل BIRTHDATE

```
USE STUDENTS
INDEX ON DESCEND(BIRTHDATE) TO B_DATE
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

FIND – INDEX – LOCATE – SEEK

الوظيفة DISKSPACE()

تظهر إجمالي المساحة المتبقية على القرص المختار.

الشكل العام:

DISKSPACE([<expN>])

حيـــث:

<expN> : رقم يدل على مشغل القرص.

الشـــرح:

تعطي هذه الوظيفة رقما يمثل إجمالي عدد الحروف التي يمكن تسجيلها على القرص المختار معك. ويجوز أن تستخدم رقما بين الأقواس ليدل على مشغل القرص إذا أردت معرفـة المساحة المتوفرة على قرص آخر فمثلاً DISKSPACE(1) تعطيك المساحة المتوفرة على القرص الموجود بمشغل القرص A ، DISKSPACE(2) لمشغل القرص B . . . وهكذا. والاستخدام الأمثل لهذه الوظيفة عندما تحتاج لعمل نسخ احتياطية من ملف كبير موجود على القرص الصلب إلى قرص أو أقراص مرنة أو عندما تحتاج لفرز ملف كبير موجود على وحدة قرص مرن لأن مساحة القرص المرن محدودة وعملية الفرز تحتاج لضعف مساحة الملف المطلوب فرزه.

ففي مثل هذه الأحوال يجب أن تتأكد أن المساحة المتبقية على القرص تسمح بنسخ أو فرز الملف قبل أن تتم عملية النسخ أو الفرز والمثال التالي يوضح لك كيف تستخدم هذه الوظيفة لتحقيق هذا الغرض.

الاختلاف عن *dBASE III PLUS* : لا تستخدم «دي بيس» الاختيار <expN> .

مثـــال:

المثـال التالي عبارة عن جزء من برنامج يستخدم لعمل نسخة احتياطية

```
FILSIZE = RECSIZE() * RECCOUNT() + HEADER()
IF DISKSPACE(1) < FILSIZE
   CLEAR
   @ 12,2 SAY "Not enough space for backing up"
   RETURN
ELSE
   RUN BACKUP C:*.DBF A:/A
ENDIF
```

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

RECCOUNT( ) – RECSIZE( ) – HEADER( )

الوظيفة DOSERROR( )

تعطي رقم آخر رسالة خطأ تأتي من نظام التشغيل DOS بسبب البرنامج.

الشكل العام:

DOSERROR( )

الشـــرح:

الأخـطاء التي يسببها البرنامج ويشعر بها نظام التشغيل DOS تحدث نتيجة استخدام أمر RUN أو عند فتح الملفات. فإذا حدث خطأ نتيجة لهذين السببين فإن الـوظيفـة DOSERROR( ) تستقبل رقما يوضح نوع الخطأ الذي حدث (ويمكنك مراجعة هذه الأرقام ومعنى كل منها في كتاب الشركة المنتجة ضمن شرح هذه الوظيفة).

ولحسن الحظ فإن معظم الأخطاء التي يشعر بها نظام التشغيل يمكن السيطرة عليها من داخل البرنامج بواسطة أوامر أو وظائف «كلبر» وعلى ذلك يمكن الاستغناء عن هذه الوظيفة.

فمثلاً الرقم 2 معناه أن الملف غير موجود (file not found) ويمكن تلافي وقوع هذا الخطأ بوضع أمر:

IF EXIST (filename)

داخل البرنامج.

وكذلك يمكن استخدام الوظيفة FERROR( ) للسيطرة على أخطاء كتابة أو قراءة الملفات.

الاختلاف عن dBASE III PLUS : غير موجودة بها.

مثـــال:

المثال التالي يظهر رسالة للمستفيد في حالة الحصول على الرسالة رقم ٢ من DOS ومعناها أن الملف غير موجود.

```
IF DOSERROR() = 2
  ? "File not available"
  WAIT " Press any key to return to main menu"
  CLOSE DATABASES
  RETURN
ENDIF
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

FERROR( ) – FOPEN( )

الوظيفة DOW()

تعطى رقم يدل على ترتيب اليوم في الأسبوع .

الشكل العام:

DOW(<*expD*>)

حيـــث:

<*expD*> : حقل أو عبارة تاريخية .

الشـــرح:

تعطى هذه الوظيفة رقم يدل على ترتيب اليوم داخل الأسبوع أي رقم من ١ إلى ٧ وهي عدد أيام الأسبوع باعتبار أن يوم الأحد هو اليوم رقم ١ .
ويمكن أن يكون <*expD*> بيانات حقل تاريخي أو تاريخ الحاسب أو تاريخ مخزن بالذاكرة .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــال:

لاظهار السجلات التي تخص يوم الأحد (اليوم الأول من الأسبوع) بفرض أن حقل DATE مسجل به بيانات تاريخية .

```
LIST FOR DOW(DATE) = 1
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

CDOW( ) – DAY( )

الوظيفة DTOC()

تحول بيانات حرفية إلى بيانات تاريخية.

الشكل العام:

DTOC(<*expD*>)

حيث:

<*expD*> : عبارة تاريخية.

الشرح:

تحول هذه الوظيفة بيانات تاريخية إلى بيانات حرفية. ويمكن الاستفادة من ذلك في حالة مقارنة بيانات حرفية مع بيانات تاريخية.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثال:

المثال الآتي يضع تاريخ اليوم بالذاكرة كبيانات حرفية.

```
STORE DTOC(DATE()) TO NEWDATE
? TYPE("NEWDATE")            && C  الإفادة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

CTOD( ) – SET CENTURY – SET DATE

الوظيفة DTOS( )

تحول تاريخ أو عبارة تاريخية إلى عبارة حرفية وتسمح بدمجها مع عبارة حرفية عند الحاجة للفهرسة باستخدام بيانات أكثر من حقل.

الشكل العام:

DTOS(<*expD*>)

حيـــث:

<*expD*> : أي تاريخ أو عبارة تاريخية.

الشـــرح:

تقبل هذه الوظيفة تاريخ أو عبارة تاريخية وتقوم بتحويله إلى عبارة حرفية وبذلك يسهل دمجه مع عبارة حرفية أخرى عند استخدام أمر INDEX ON . وتأخذ العبارة التاريخية بعد تحويلها إلى عبارة حرفية هذا الشكل YYYY MM DD بصرف النظر عن وضع أمر SET DATE وأمر SET CENTURY فمثلاً: التاريخ: 10/19/90 بعد تحويله إلى عبارة حرفية يصير هكذا: "19901019" .

والفائدة من ذلك إمكانية دمج التاريخ مع العبارة الحرفية عند الحاجة للفهرسة باستخدام بيانات أكثر من حقل، انظر المثال التالي:

```
INDEX ON LASTNAME + DTOS (BIRTHDATE) TO NAME_DATE
```

في هذا المثال فإن العبارة المتخذة أساساً لترتيب الملف هي LASTNAME + DTOS(BIRTHDATE) . ولا تصلح الوظيفة DTOC( ) في هذا المثال. لأن الوظيفة DTOC( ) تحول التاريخ إلى عبارة حرفية بهذا الشكل "MM DD YY" . وفي هذه الحالة فإن مواليد شهر يناير بصرف النظر عن سنة الميلاد سيتم ترتيبهم بجوار بعض في أول الملف. وهذا غير المقصود.

الاختلاف عن *dBASE III PLUS* : غير موجودة بها.

مثـــال:

```
USE STUDENTS
INDEX ON LASTNAME + DTOS(BIRTHDATE) TO NAM_DAT
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

CTOD( ) - DTOC( )

الوظيفة DTOS( )

تحول تاريخ أو عبارة تاريخية إلى عبارة حرفية وتسمح بدمجها مع عبارة حرفية عند الحاجة للفهرسة باستخدام بيانات أكثر من حقل.

الشكل العام:

DTOS(<*expD*>)

حيث:

<*expD*> : أي تاريخ أو عبارة تاريخية.

الشرح:

تقبل هذه الوظيفة تاريخ أو عبارة تاريخية وتقوم بتحويله إلى عبارة حرفية وبذلك يسهل دمجه مع عبارة حرفية أخرى عند استخدام أمر INDEX ON . وتأخذ العبارة التاريخية بعد تحويلها إلى عبارة حرفية هذا الشكل YYYY MM DD بصرف النظر عن وضع أمر SET DATE وأمر SET CENTURY فمثلاً: التاريخ: 10/19/90 بعد تحويله إلى عبارة حرفية يصير هكذا: "19901019"

والفائدة من ذلك إمكانية دمج التاريخ مع العبارة الحرفية عند الحاجة للفهرسة باستخدام بيانات أكثر من حقل، انظر المثال التالي:

INDEX ON LASTNAME + DTOS (BIRTHDATE) TO NAME_DATE

في هذا المثال فإن العبارة المتخذة أساساً لترتيب الملف هي LASTNAME + DTOS(BIRTHDATE) . ولا تصلح الوظيفة DTOC( ) في هذا المثال. لأن الوظيفة DTOC( ) تحول التاريخ إلى عبارة حرفية بهذا الشكل "MM DD YY" . وفي هذه الحالة فإن مواليد شهر يناير بصرف النظر عن سنة الميلاد سيتم ترتيبهم بجوار بعض في أول الملف. وهذا غير المقصود.

الاختلاف عن *dBASE III PLUS* : غير موجودة بها.

مثـــال:

```
USE STUDENTS
INDEX ON LASTNAME + DTOS(BIRTHDATE) TO NAM_DAT
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

CTOD( ) – DTOC( )

الوظيفة EMPTY( )

تحدد هل تعبير ما يشتمل على فراغات أو صفر أو القيمة .F.

الشكل العام:

EMPTY(<*exp*>)

حيـــث:

<*exp*> : أي تعبير صحيح أو حقل بالذاكرة.

الشـــرح:

تعطي هذه الوظيفة القيمة المنطقية .T. إذا كانت العبارة أو الحقل المشار إليه لا يشتمل على بيانات. ويحدث ذلك الأحوال الآتية:

- إذا كان حقل أو عبارة حرفية تشتمل كلها على فراغات.
- إذا كان حقل أو عبارة رقمية تشتمل على صفر.
- إذا كان حقل أو عبارة تاريخية لا يشتمل على شيء (أي يشتمل على " //").
- إذا كان حقل أو عبارة منطقية تشتمل على .F.
- إذا كان حقل أو عبارة ملاحظات يشتمل على فراغات.

الاختلاف عن *dBASE III PLUS* : غير موجودة بها.

مثـــال:

```
ACCEPT "Enter student number or press Enter to exit" TO MNO
IF EMPTY(MNO)
   RETURN
ENDIF
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

LEN( )

الوظيفة EOF()

توضح هل يقف المؤشر عند نهاية الملف أم لا.

الشكل العام:

EOF( )

الشـــرح:

تستخدم هذه الوظيفة في الغالب داخل البرامج لمعرفة نهاية سجلات الملف. وتعطي هذه الوظيفة القيمة المنطقية .T. بمعنى صح (True) إذا كان المؤشر وصل إلى نهاية الملف. وتعتبر قاعدة البيانات الوظيفية EOF( ) صحيحة إذا كان المؤشر يقف بعد آخر سجل بسجل.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

المثال الآتي جزء من برنامج يستخدم هذه الوظيفة لتحديد نهاية سجلات الملف.

```
USE STUDENTS
DO WHILE .NOT. EOF()
  ? "Student no.     " + STUDENTNO
  ? "Student name    " + TRIM(FIRSTNAME) + LASTNAME
  ? "Student address" + TRIM(ADDRESS) + "," + TRIM(CITY) + "."
  ?
  SKIP
ENDDO
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

BOF( ) – FOUND( )

الوظيفة EXP( )

تعطى قيمة لوغاريتم الرقم ١ مرفوعا لقوة معينة.

الشكل العام:

EXP(<expN>)

حيـــث:

<expN> : أي رقم أو عبارة رقمية.

الشـــرح:

هذه الوظيفة عكس وظيفة LOG( ) وهي تعطى قيمة الثابت e * مرفوعا للقوة المماثلة للرقم المحدد في الاختيار <expN>.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

لايجاد الثابت e مرفوعا للقوة ١

```
SET DECIMALS TO 2
? EXP(1)                  && 2.72   الاجابة
```

ولايجاد قيمة الثابت e مرفوعا للقوة ٢

```
? EXP(2)                  && 7.39   الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET DECIMALS – SET FIXED

* الحرف e يمثل قيمة ثابتة قدرها ٢,٧١٨٢٨ ١٨٢٨٥ وتعادل لوغاريتم الرقم ١.

الوظيفة FCLOSE( )

تحفظ ملف نصي من الذاكرة إلى القرص المغنط سبق فتحه بالوظيفة FCREATE( ) .

الشكل العام:

FCLOSE(<*expN*>)

حيـــث:

<*expN*> : عبارة رقمية تنتج من استخدام الوظيفة FCREATE( ) أو الوظيفة FOPEN( ) .

الشـــرح:

هذه الوظيفة من الوظائف المتقدمة التي لا يستخدمها إلا المبرمجين الذين يعرفون كيف يتعامل نظام التشغيل مع الملفات. لأن الملف الذي تغلقه يكون موجوداً بالمحطة الانتقالية (Buffer) داخل الذاكرة. وعادة تكون FCLOSE( ) آخر إجراء بعد إنشاء الملف بالوظيفة FCREATE( ) وفتحه بالوظيفة FOPEN( ) .

وتشتمل هذه الوظيفة على القيمة .F. إذا حدث خطأ أثناء كتابة الملف. وإلا فإنها تشتمل على القيمة .T.

الاختلاف عن *dBASE III PLUS*: غير موجودة بها.

مثـــال:

المثال التالي يغلق الملف الذي سبق إنشاؤه بالذاكرة ونقله إلى القرص المغنط.

```
MHAND = FCREATE("ASCFIL.TXT",0)   && انشىء ملفا جديدا (قراءة وكتابة)
IF MHAND = - 1                    && اذا لم يكن الملف موجودا
  ? " Unable to create file"
  RETURN
ENDIF
MTITL = "Kingdom of Saudi Arabia"
FWRITE(MHAND,MTITLE)              && ادخل ملف ASCFIL.TXT
                                  && من الذاكرة الى القرص &&
FCLOSE(MHAND)                     && اغلق الملف
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

FCREATE( ) – FWRITE( ) – FOPEN( )

الوظيفة FCOUNT( )

تحسب عدد الحقول في الملف المفتوح.

الشكل العام:

FCOUNT( )

الشـــرح:

تعطي هذه الوظيفة رقماً يدل على عدد الحقول في الملف المفتوح فإذا لم تجد ملفاً مفتوحاً فإنها تعطي الرقم 0 .

الاختلاف عن dBASE III PLUS : غير موجودة بها.

مثـــال:

المثال التالي يظهر أسماء الحقول الموجودة بالملف المفتوح.

```
FOR I = 1 TO FCOUNT()
  ? FIELD(I)
NEXT
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

AFIELD( ) – FIELD( ) – FIELDNAME( )

FCREATE( ) الوظيفة

تنشيء ملف نصي داخل الذاكرة أو لاستخلاص محتويات ملف بغد تركه بدون بيانات.

الشكل العام:

FCREATE (<expC> [,<expN>])

حيث:

<expC> : اسم الملف المطلوب إنشاؤه.

<expN> : رقم يوضح حالة الملف المطلوب إنشاؤه سنوضحه بعد قليل.

الشرح:

تنشىء هذه الوظيفة ملفاً نصياً مكتوباً بشفرة ASCII ويجوز إضافة رقم بعد اسم الملف ليبين حالة الملف المطلوب إنشاؤه وإليك بيان بالأرقام المسموحة ومعنى كل منها:

الرقم	حالة الملف الذي ينشئه
0	ملف عادي (Normal)
1	ملف قراءة فقط (Read only)
2	ملف مخفي (Hidden)
3	ملف قراءة فقط مخفي (Read and hidden)
4	ملف للنظام (System)

فإذا لم تستخدم هذا الرقم ضمن الأمر فسيتم إنشاء ملف عادي وإذا استخدمت اسم ملف موجود من قبل فسيمحى الملف القديم. ولذلك يستحسن في

هذه الحالة أن تستخدم الوظيفة ( )FOPEN .

لاحظ أن الملف الذي سيتم إنشاؤه سيوضع على الدليل الحالي لأن هذه الوظيفة تتجاهل أمر SET PATH وأمر SET DEFAULT .

وعادة يخصص نظام التشغيل رقماً للملف عندما يتم فتحه يسمى هذا الرقم File handle وعادة تشتمل الوظيفة ( )FCREATE على هذا الرقم وإذا لم يتم إنشاء الملف بهذه الوظيفة فستشتمل الوظيفة على الرقم 1- . فإذا أردت التعامل مع هذا الرقم فيما بعد لأغراض قراءة أو كتابة أو فتح الملف ضع نتيجة الوظيفة ( )FCREATE داخل حقل ذاكرة (Memory Variable) هكذا :

MHAND = FCREATE ("ASCFIL.TXT")

ملاحظة : الوظائف التي تبدأ بحرف F مثل

FCREATE( ) – FOPEN( ) – FWRITE( ) – FCLOSE( ) –

FERROR( ) – FREAD( ) – FREADSTR( ) – FSEEK( )

يحتاج إليها المبرمج المتمرس للتعامل مع ملفات نصية (ملفات غير «دي بيس») .

الاختلاف عن dBASE III PLUS : لا توجد بها.

مثـــال:

المثال التالي ينشىء ملف قراءة وكتابة باسم ASCFIL.TXT على الدليل الحالي وطول هذا الملف هو طول العبارة "Kingdom of Saudi Arabia"

```
MHAND = FCREATE("ASCFIL.TXT",0)   && انشىء ملفا جديدا(قراءة وكتابة)
IF MHAND = - 1                    && اذا لم يتم انشاء الملف
  ? " Unable to create file"
  RETURN
ENDIF
MTITL = "Kingdom of Saudi Arabia"
FWRITE(MHAND,MTITLE)              && ASCFIL.TXT ادخل ملف
                                  &&من الذاكرة الى القرص
FCLOSE(MHAND)                     && اغلق الملف
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

FCLOSE( ) – FOPEN( )

الوظيفة FERROR( )

تحدد نوع الخطأ الذي حدث نتيجة استخدام إحدى وظائف الملفات النصية .

الشكل العام:

FERROR( )

الشـــــرح:

المقصود بوظائف الملفات النصية الوظائف التي تنشيء أو تفتح أو تكتب أو تغلق ملف نصي وهي الوظائف التي تبدأ بحرف F إذا حدث خطأ نتيجة لاحدى هذه الوظائف فإن الوظيفة FERROR( ) ستشتمل على رقم معين (ويمكنك مراجعة كتاب الشركة المنتجة لمعرفة هذه الأرقام ومعنى كل منها) . وإذا لم يحدث خطأ فإن هذه الوظيفة ستشتمل على الرقم صفر.

الاختلاف عن *dBASE III PLUS* : غير موجودة بها.

مثـــال:

المثـال التالي يفتح ملف نصي اسمه ASCFIL.TXT فإذا حدث خطأ (إذا اشتملت الوظيفة FERROR( ) على رقـم أكثر من صفر) يظهر رسالة خطأ تشتمل على رقم الخطأ الذي حدث.

```
MHAND = FOPEN("ASCFIL.TXT",0)       && ASCFIL.TXT افتح ملف
                                    && لغرض القراءة والكتابة
IF FERROR() <> 0                    && اذا حدث خطأ
   CLEAR
   * Display an error message including error number
   @ 12,2 SAY " Error in opening fil. Error No."+ STR(FERROR(),2)
   WAIT " Press a key to exit "
   CLEAR
   CLEAR ALL
   RETURN
ENDIF
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

FCREATE( ) – FOPEN( ) – FWRITE( ) – FSEEK( )

الوظيفة FIELD( )/FIELDNAME( )

تعطي اسم حقل داخل الملف المفتوح.

الشكل العام:

FIELD (<*expN*> / FIELDNAME (<*expN*>)

حيـــث:

<*expN*> : رقم أو تعبير رقمي.

الشــــرح:

تعطي هذه الوظيفة اسم الحقل المشار إلى رقمه (<*expN*>) داخل ملف قاعدة البيانات المفتوح. وتعتبر قاعدة البيانات أن الحقل الأول داخل بناء الملف رقمه ١ - FIELD(1) - والثاني رقمه ٢ . . . وهكذا. . .

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثـــال:

المثال الآتي يستخدم ملف STOCK.dbf لمعرفة ثالث حقل بالملف.

```
USE STOCK
STORE 3 TO FLD_NO
? FIELD(FLD_NO)
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

DBF( ) – RECCOUNT( ) – RECSIZE( )

الوظيفة FILE( )

تعطي الاجابة .T. إذا كان الملف موجودا على الدليل الحالي وإلا .F.

الشكل العام:

FILE(<*filename*>)

حيـــث:

<*filename*> : اسم الملف المطلوب الاستفسار عنه.

الشـــرح:

تستخدم هذه الوظيفة للتأكد من أن الملف المذكور موجود أم لا على الدليل الحالي وفي حالة العثور على اسم الملف فإن الوظيفة تعطي الاجابة .T. بمعنى True أو صح وإلا فتعطي .F. بمعنى False أو خطأ.

ويجب أن يشتمل اسم الملف على الاسم الأصلي والاسم الممتد وإذا كان الملف موجودا على دليل أو قرص آخر غير المخصص معك فيجب أن يسبق اسم الدليل أو مشغل القرص اسم الملف.

إذا لم يكن اسم الملف موجـودا بالـذاكـرة فيجب أن يكتب بين علامتي تنصيص " ".

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

المثـال الآتي يتـأكد من وجود ملف TMPSALE.dbf على مشغل القرص B بفرض أن مشغل الوحدة المختار C لأنه سيشتمل على فواتير البيع قبل تسجيلها لملف المبيعات. فإذا لم يكن موجودا فإن البرنامج ينشئه تلقائيا ليكون جاهزا.

```
STORE "B:TEMPSALE.DBF" TO TEMP
IF .NOT. FILE(TEMP)
    USE SALE
    COPY STRUCTURE TO B:TEMPSALE
ENDIF
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

DIR – SET PATH

الوظيفة FLOCK( )

تحاول غلق الملف المفتوح حتى لا يتمكن الآخرون من الكتابة عليه.

الشكل العام:

FLOCK( )

الشـــرح:

تستخدم هذه الوظيفة في حالة استخدام شبكة اتصالات محلية وهي تحاول غلق الملف المفتوح لاجراء تعديلات على محتوياته أو لأي سبب آخر حتى لا يتأثر بتعديلات الآخرين التي تتم في نفس اللحظة. فإذا نجحت في غلق الملف فإنها تشتمل على القيمة المنطقية .T. ويبقى الملف مغلقاً حتى يصدر الشخص الذي أغلقه أمر UNLOCK أو يغلقه تماماً.

لاحظ أن هذه الوظيفة تسمح للآخرين داخل نفس الشبكة بالاطلاع على الملف فقط بدون صلاحيات التعديل فيه. فإذا أردت منع الآخرين حتى من قراءة الملف استخدم أمر USE...EXCLUSIVE.

الاختلاف عن dBASE III PLUS: تستخدم مع «دي بيس» لتمنع باقي المستفيدين من القراءة أو الكتابة على الملف.

مثـــال:

المثال التالي يتأكد أن الملف مغلق قبل أن يبدأ طباعة التقرير.

```
IF FLOCK()
   REPORT FORM STKRPRT TO PRINT
ENDIF
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

USE...EXCLUSIVE – UNLock – SET EXCLUSIVE – RLOCK( )

الوظيفة FOPEN()

تفتح ملف نصي أنشىء بالوظيفة FCREATE() .

الشكل العام:

FOPEN (<*expC*> [,<*expN*>])

حيـــث:

<*expC*> : اسم الملف المطلوب فتحه.

<*expN*> : رقم يوضح حالة الملف (صفر: تعني القراءة فقط، ١: تعني الكتابة فقط، ٢: تعني القراءة والكتابة).

الشـــرح:

هذه الوظيفة من الوظائف المتقدمة التي يستخدمها المبرمج المتمرس للتعامل مع ملفات نصية (غير «دي بيس») وهي تقوم بفتح ملف أنشىء بالوظيفة FCREATE() ويجب أن يشتمل اسم الملف المطلوب فتحه على اسم الدليل ومشغل القرص إذا كان موجوداً على دليل آخر إذا لم يشتمل الأمر على <*expN*> فإن «كلبر» ستخصص الرقم صفر ويعني فتح الملف لغرض القراءة فقط.

وعادة يخصص نظام التشغيل رقماً للملف عندما يتم فتحه يسمى هذا الرقم File . ويوضـع هذا الرقم داخل هذه الوظيفة. أما إذا لم يفتح الملف فستشتمل الوظيفة على الرقم 1- فإذا أردت التعامل مع هذا الرقم فيما بعد لأغراض القراءة أو الكتابة ضع نتيجة الوظيفة داخل حقل ذاكرة هكذا:

MHAND = FOPEN("ASCFIL.TXT")

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال :

المثال التالي يفتح الملف ASCFIL.TXT للقراءة فقط .

```
MHAND = FOPEN("ASCFIL.TXT",0)     && افتح الملف للقراءة فقط
* < Command to read or write to file>
FCLOSE(MHAND)                     && اغلق الملف
```

المكتبة : EXTEND.LIB

الأوامر ذات الصلة :

FCLOSE( ) – FCREATE( ) – FWRITE( ) – FERROR( )

الوظيفة FOUND()

تحدد هل وجدت قاعدة البيانات السجل المطلوب أم لا .

الشكل العام:

FOUND( )

الشـــرح:

تستخـدم هذه الـوظيفـة داخـل البرنامج لمعرفة هل وجدت قاعدة البيانات السجل المطلوب البحث عنه داخل الملف أم لا . ويتم البحث داخل الملف بأحد أوامر البحث المعروفة وهي :

LOCATE – FIND – SEEK – CONTINUE

فإذا كان السجل موجودا فإن الوظيفة تعطي القيمة المنطقية .T. بمعنى صح (True) وإذا لم يكن موجودا فستعطي .F. بمعنى خطأ (False) .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

المثال التالي يستخدم ملف STUDENTS.dbf للبحث عن مدينة الرياض (CITY = "RIYADH") بفرض أن الملف سبق فهرسته طبقا لبيانات حقل CITY .

```
USE STUDENTS INDEX ICITY
SEEK "RIYADH"
IF FOUND()
   ? CITY
ELSE
   ? " Ciry name not found"
ENDIF
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

LOCATE – CONTINUE – SEEK – FIND – EOF( )

الوظيفة FREADSTR()

تقرأ جزء من ملف نصي مفتوح.

الشكل العام:

FREADSTR (<*expN1*> , <*expN2*>)

حيــث:

<*expN1*> : الرقم الذي نحصل عليه من الوظيفة FOPEN( ) أو FCREATE( )

<*expN2*> : رقم يحدد عدد الحروف المطلوب قراءتها من الملف النصي.

الشــرح:

تستطيع هذه الوظيفة قراءة حروف يصل عددها إلى ٦٤ك.ب. إذا لم تجد هذه الوظيفة بيانات في الملف المذكور أو إذا وصلت إلى نهاية الملف قبل قراءة عدد الحروف المحددة فستحصل على خطأ.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثــال:

المثال التالي يستخدم هذه الوظيفة لقراءة الحروف الستة عشر الأولى من ملف نصي لإجراء بعض العمليات عليها.

```
MHAND = FOPEN("ASCFIL.TXT",0)      && افتح الملف للقراءة فقط read only
IF FERROR <> 0                     && اذا حدث خطأ
   @ 12,10 SAY " Error in opening file"
   WAIT " Press a key to return"
   RETURN
ELSE
   MBUFFER = FREADSTR(MHAD,16)      && اقرأ 16 حرفا الاولى من الملف
   * < Command to use the text file >
   FCLOSE(MHAND)                    && اغلق الملف
ENDIF
```

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

FERROR( ) – FOPEN( ) – FCREATE( ) – FCLOSE( )

الوظيفة FWRITE()

تكتب عبارة بالذاكرة على ملف نصي.

الشكل العام:

FWRITE (<*expN1*>, <*memvarC*>[,<*expN2*>])

حيث:

<*expN1*> : الرقم الذي نحصل عليه من الوظيفة FOPEN( ) أو FCREATE( )

<*memvarC*> : عبارة مخزنة بالذاكرة.

<*expN2*> : عدد الحروف المطلوب كتابتها من الذاكرة.

الشرح:

تقوم هذه الوظيفة بكتابة عبارة موجودة بالذاكرة على ملف نصي. وتشتمل على عدد الحروف التي كتبت، إذا لم تشتمل الوظيفة على الاختيار <*expN2*> فستتم كتابة العبارة كلها.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثال:

المثال التالي يكتب عبارة Cairo على الملف النصي المفتوح.

```
MCITY = "Cairo"
MHAND = FOPEN("ASCFIL.TXT",0)       && افتح الملف للقراءة فقط
IF FERROR <> 0                      && اذا حدث خطأ اثناء فتح الملف
   @ 12,10 SAY " Error in opening file. ERROR "+STR(FERROR())
   WAIT " Press a key to return"
   RETURN
ELSE
   FWRITE(MHAND,MCITY)                && ASCFIL.TXT اكتب على ملف
   IF FERROR <> 0                     && اذا حدث خطأ اثناء الكتابة
     @ 12,10 SAY " Cann't write. ERROR "+STR(FERROR())
     WAIT " Press a key to return"
     RETURN
   ENDIF
   FCLOSE(MHAND)                      && اغلق الملف
ENDIF
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

FCREATE( ) – FCLOSE( ) – FERROR( ) – FOPEN( ) – FREADSTR( )

الوظيفة GETE()

تعطي معلومات عن نظام التشغيل.

الشكل العام:

GETE (<*expC*>)

حيـــث:

<*expC*> : أحد متغيرات نظام التشغيل مثل PATH أو COMSPEC

الشــــرح:

المقصـود بمتغيرات نظام التشغيل المتغيرات التي تنشأ في الذاكرة والتي نقول عنها تجاوزا حقول ذاكرة. ويتم تعريفها لنظام التشغيل إما من محث DOS أو في ملف تجميعي (Bateh file) مثـل PATH أو PROMPT أو SET . فإذا أردت أن تظهـر معلومات عن نظام التشغيل الذي تستخدمه استخدم هذه الوظيفة. إذا لم تجد «كلبر» معلومات تخص المتغير الذي تسأل عنه فلن يظهر لك شيء.

الاختلاف عن ***dBASE III PLUS*** : تستخدم «دي بيس» الوظيفة GETENV() بدلاً منها.

مثــــال:

```
? GETE ("PATH")         && ستحصل على اجابة بهذا الشكل
                        && C:\DOS;C:\CLIPPER;C:\DBASE
? GETE ("COMSPEC")      && ستحصل على اجابة بهذا الشكل
                        && C:\COMMAND.COM
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة: لا توجــد.

الوظيفة HEADER( )

تحسب عدد الحروف المخصصة كعناوين للملف المفتوح.

الشكل العام:

HEADER( )

الشـــرح:

تخصص مساحة لكل ملف من ملفات قاعدة البيانات يكتب فيها معلومات عن هذا الملف مثل نوع الملف وعدد السجلات وتاريخ آخر تعديل تم على الملف بالاضافة إلى معلومات عن حقول الملف مثل أطوالها وأنواعها وأسمائها هذه المعلومات هي التي يطلق عليها Header . فإذا أردت أن تعرف المساحة المخصصة لهذه العناوين استخدم الوظيفة HEADER( ) ونحتاج لهذه الوظيفة إذا أردنا أن نعرف المساحة التي يشغلها الملف على القرص لأغراض نقل الملف على قرص آخر أو ما شابه ذلك. وتحسب المساحة التي يشغلها الملف على القرص بضرب عدد السجلات × حجم السجل + مساحة العناوين (Header) .

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال التالي يحسب المساحة المتوفرة على القرص المرن قبل نقل ملفات DBF. إليه. فإذا كانت تكفي تمت عملية النقل. وإلا أظهر رسالة للمستفيد ورجع إلى البرنامج الرئيسي.

```
* File size = (no. of records * record size) + header information
FILSIZE = RECSIZE() * RECCOUNT() + HEADER()
IF DISKSPACE(1) < FILSIZE
   CLEAR
   @ 12,2 SAY "Not enough space for backing up"
   RETURN
ELSE
   RUN BACKUP C:*.DBF A:/A
ENDIF
```

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

DISKSPACE( ) – LASTREC( ) – RECSIZE( ) – RECCOUNT( )

٥ الوظيفة IF( )/IFF( )

تقيم الحالة المشروحة في الوظيفة (صح أو خطأ) وتختار احدى عبارتين بناء على نتيجة التقييم.

الشكل العام:

IIF(<*expL*>, <*exp1*>, <*exp2*>)

حيـــث:

<*expL*> : أي تعبير منطقي له إحدى نتيجتين صح أو خطأ.

<*exp1*> : العبارة المختارة إذا وقعت نتيجة المقارنة صحيحة.

<*exp2*> : العبارة المختارة إذا وقعت نتيجة المقارنة خاطئة.

الشـــرح:

هذه الوظيفة تقوم بعمل أمر IF...ELSE...ENDIF ولكنها لا تستخدم لتنفيذ أمر أو مجموعة أوامر.

وإنما هي تقيم تعبيرا منطقيا يحتمل الصواب والخطأ ويعبر عنه في الأمر بالاختيار <*expL*> فإذا كانت النتيجة صواب فإنها تعطي العبارة الأولى (<*exp1*>) أما إذا كانت النتيجة خطأ فإنها تعطي العبارة الثانية (<*exp2*>).

لاحظ أن كلا من <*exp1*> و <*exp2*> يمكن أن يكون تعبيرا حرفيا أو رقميا أو تاريخيا.

لاحظ أن استخدام هذه الوظيفة كبديل لأمر:

IF...ELSE...ENDIF

داخل البرنامج يزيد من سرعة تنفيذ البرنامج.

الاختلاف عن dBASE III PLUS : مع «كلبر» لا يشترط أن يكون كل من <*exp1*> و <*exp2*> من نفس النوع.

مثـــال:

إذا قررت أن يظهر الرقم ١٠٠ كحد أدنى للأسعار (PRICE) في ملف STOCK.dbf فيمكنك استخدام الأمر التالي:

```
. LIST IIF(PRICE < 100, 100, PRICE)
```

وفي هذا المثـال إذا كان السعـر مساويا أو أقل من الرقم ١٠٠ فإن المقارنة صحيحة وسيظهر الرقم ١٠٠ بدلا من السعر المسجل بالملف أما إذا كان السعر أكبر من ١٠٠ فإن المقارنة خاطئة. وسيظهر الرقم الموجود بحقل PRICE .

لكي تظهر رسالة "Good morning" للمشغل إذا بدأ العمل في الصباح قبل الساعة الثانية عشرة ظهرا وإلا "Good afternoon" .

```
MSG = IIF(TIME() < "12:00:00", "Good morning", "Good afternoon")
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

IF...ENDIF

الوظيفة INDEXEXT( )

تحدد هل تم ربط ملف NDX.OBJ مع النظام أم لا؟.

الشكل العام:

INDEXEXT( )

الشـــرح:

من المعروف أن الاسم الممتد لملف الفهرس الذي تستخدمه «دي بيس ثري بلاس» هو "NDX." وهو يصلح أيضا للعمل مع «كلبر». وأن «كلبر» يستخدم ملف "NTX." ولكن «دي بيس» لا تستطيع استخدامه.

فإذا كان ملف الفهرس من نوع NTX. فإن هذه الوظيفة ستعطيك عبارة "NTX" أما إذا كان من نوع NDX. فستعطيك عبارة "NDX" .

ويستفاد من هذه الوظيفة في معرفة نوع الفهرسة لتحديد هل يمكن تشغيل هذا النظام باستخدام «دي بيس ثري بلاس» أم لا؟ فإذا كان نوعه NDX. فالاجابة صحيحة.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال التالي يستخدم أمر IF لمعرفة نوع ملف الفهرس لتحديد هل يمكن تشغيل النظام باستخدام «دي بيس ثري بلاس» أم لا؟.

```
IF INDEXEXT() = "NTX"
  ? "Cann't run with dBASE III PLUS"
ELSE
  ? "dBASE III PLUS compatible"
ENDIF
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

INDEXKEY( ) – INDEXORD( )

الوظيفة INDEXKEY( )

تعطي اسم الحقل أو العبارة المتخذة كأساس لترتيب ملف الفهرس.

الشكل العام:

INDEXKEY(<*expN*>)

حيـــث:

<*expN*> : ترتيب ملف الفهرس داخل أمر USE...INDEX أو SET INDEX TO

الشــــرح:

تعطي الوظيفة اسم الحقل أو العبارة التي تم على أساسها ترتيب الملف ويجب أن تحدد للوظيفة رقماً يدل على الترتيب المطلوب فإذا زاد الرقم الذي حددته عن عدد الفهارس المفتوحة فلن تحصل على شيء إذا أردت الاشارة إلى الملف الرئيسي الذي يتحكم في ترتيب الملف عند فتح أكثر من فهرس استخدم الرقم صفر بدلاً عن expN .

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال التالي يفتح ملف STOCK.DBF ومعه ٣ فهارس ثم يسأل بعد ذلك عن هذه الفهارس وعن الفهرس الرئيسي.

```
USE STOCK INDEX ITEMNO,DESC,PRICE
SET ORDER TO 2
KEY1 = INDEXKEY(1)
KEY2 = INDEXKEY(2)
KEY3 = INDEXKEY(3)
PRIMKEY = INDEXKEY(0)
*
? KEY1              && ITEMNO  الإجابة
? KEY2              && DESC    الإجابة
? KEY3              && PRICE   الإجابة
? INDEXKEY(5)       && لن تحصل على شيء
? PRIMKEY           && DESC    الإجابة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

USE – SET INDEX – SET ORDER – INDEXORD( ) – INDEXEXT( )

الوظيفة INDEXORD()

تحدد ملف الفهرس المتخذ كأساس لترتيب الملف من بين ملفات الفهرس المفتوح .

الشكل العام:

INDEXORD( )

الشـــرح:

تعطي هذه الوظيفة رقما يدل على ملف الفهرس المتخذ كأساس لترتيب ملف قاعدة البيانات عند فتح أكثر من فهرس بأمر USE...INDEX أو SET INDEX . فإذا لم تجد «كلبر» ملف فهرس مفتوحا فستعطيك الرقم صفر.

الاختلاف عن *dBASE III PLUS* : لا توجد بها .

مثـــال:

```
USE STOCK INDEX ITEMNO,DESC,PRICE
? INDEXORD()            && 1  الاجابة
SET ORDER TO 2
? INDEXORD()            && 2  الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

USE – INDEX – SET ORDER – SET INDEX

الوظيفة INKEY()

تسمح للبرنامج أن يقرأ أحد حروف لوحة المفاتيح عند إدخال بيانات بواسطة المستخدم.

الشكل العام:

INKEY()

الشـــرح:

تعطي هذه الوظيفة رقما صحيحا يمثل آخر حرف تم الضغط عليه من لوحة المفاتيح وهذا الرقم يقع بين صفر و٢٥٥ وهي الأرقام المقابلة لحروف لوحة المفاتيح في الشفرة الأمريكية (ASCII Code) . وهذا يعني أنها تعطي الرقم المقابل لمفاتيح الحروف الخاصة مثل مفاتيح الأسهم أو مفاتيح Ctrl-End أو مفتاح Esc . . . الخ .

وهذا يسمح بتوظيف مفاتيح هذه العلامات والرموز الخاصة والتي لا تظهر على الشاشة لأغراض التفريع في حالة الاختيار بين أكثر من حالة.

يوضح لك الجدول التالي الرقم المقابل لمجموعة الحروف والعلامات الخاصة في حالة استخدام الوظيفة INKEY .

المفتاح Keys	الرقم المقابل للوظيفة INKEY( )
→	4
←	19
↑	5
↓	24
Ctrl →	2
Ctrl ←	26
Ins	22
Del	7

المفتاح Keys	الرقم المقابل للوظيفة INKEY()
Home	1
End	6
PgUp	18
PgDn	3
Ctrl-Home	29
Ctrl-End	23
Esc	27
Ctrl-PgUp	31
Ctrl-PgDn	30

لاحظ أن هذه المفاتيح إذا استخدمت في حالة الاظهار فستعطي نتيجة مختلفة فمثلا إذا أدخلت هذا الأمر:

?CHR(4)

فستحصل على رمز آخر غير السهم المتجه لليمين → وذلك لأن هذه الأكواد الموضحة أمام العلامات الموجودة في الجدول تأخذ هذه القيم مع الوظيفة INKEY() فقط.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال ١ :

المثال التالي يستخدم الوظيفة INKEY() لاظهار الوقت ثانية ثانية حتى يتم ضغط أحد المفاتيح.

```
i = 0
DO WHILE I = 0
   I = INKEY()
   @ 1,58 SAY "Date:" + DTOC(DATE())
   @ 2,58 SAY "Time:" + TIME()
   @ 2,72 SAY CHR(I)
ENDDO
```

مثـــال ٢ :

المثال التالي يطبع بيانات ملف STOCK.DBF ويتيح للمستخدم إنهاء البرنامج متى أراد بالضغط على مفتاح C (Cancel) ولذلك وضعنا الوظيفة INKEY( ) داخل دوارة لتتعرف على آخر حرف يتم الضغط عليه.

```
USE STOCK
SET PRINT ON
DO WHILE .NOT. EOF() .AND. INKEY() # 67  && 67 Is the ASCII for C
   ?
   ? "Item No....... " + ITEM_NO
   ? "Description ..." + DESC
   ? "Quantity ......" + LTRIM(STR(ON_HAND))
   SKIP
   ?
ENDDO
SET PRINT OFF
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة:

ON KEY – READ – SET TYPEHEAD – CHR( ) – READKEY( )

الوظيفة INT( )

تعطيك الرقم الصحيح من قيمة عشرية.

الشكل العام:

INT(<expN>)

حيـــث:

<expN> : أي رقم أو تعبير رقمي.

الشـــرح:

تحول أي رقم أو تعبير رقمي موجود بالملف أو بالذاكرة إلى رقم صحيح. وذلك لأنها تحذف الأرقام الموجودة بعد العلامة العشرية.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

```
STORE 123.45 TO MVALU
? INT(MVALU)                  && 123  الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

ROUND( ) – MOD( )

الوظيفة ISALPHA()

تحدد هل تبدأ العبارة بحرف أبجدي أم لا.

الشكل العام:

ISALPHA(<*expC*>)

حيث:

<*expC*> : عبارة أو حقل حرفي.

الشرح:

تعطيك هذه الوظيفة نتيجة من حرف واحد هو: .T. بمعنى صح (True) أو .F. بمعنى خطأ (False) للاجابة على سؤالك: هل تبدأ العبارة الحرفية بحرف أبجدي أم لا؟

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثال:

```
ALPH1 = "Magdi100"
ALPH2 = "100Magdi"
? ISALPHA(ALPH1)          && .T. الاجابة
? ISALPHA(ALPH2)          && .F. الاجابة
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

ISLOWER( ) – ISUPPER( )

الوظيفة ISCOLOR( )

تعطي إجابة .T. إذا كان الحاسب يشتمل على شاشة ملونة وإلا .F. .

الشكل العام:

ISCOLOR( )

الشـــرح:

تعطيك هذه الوظيفة القيمة المنطقية .T. إذا كانت شاشة الحاسب ملونة أو القيمة .F. إذا كانت شاشة الحاسب غير ملونة.

وبذلك يمكن تحديد هل تستخدم ألوان عند تصميم البرامج التي ستنفذ على هذا الحاسب أم لا.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

ضع المثال الآتي في البرنامج الرئيسي قبل تنفيذ أي برنامج لتحدد الألوان التي ستستخدمها مع الشاشة الملونة.

```
IF ISCOLOR()
  SET COLOR TO W+/B+,GR+/R+
ELSE
  SET COLOR TO W+/N
ENDIF
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

SET COLOR – SETCOLOR( )

الوظيفة ISLOWER()

تحدد هل تبدأ العبارة بحرف صغير أم لا.

الشكل العام:

ISLOWER(<*expC*>)

حيث:

<*expC*> : عبارة أو حقل حرفي.

الشرح:

تستخدم هذه الوظيفة لمعرفة ما إذا كانت عبارة أو حقل حرفي تبدأ بأحد الحروف الصغيرة (Lower Case Letters) أم لا. فإذا كانت الاجابة صحيحة فستعطيك .T. بمعنى صح وإلا فستعطيك .F. بمعنى خطأ.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثال:

```
ALPH1 = "MAGDI"
? ISLOWER(ALPH1)          && .F.  الاجابة
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

ISALPHA( ) – ISUPPER( ) – LOWER( ) – UPPER( )

الوظيفة ISPRINTER( )

تحدد هل الطابعة جاهزة لتلغي المخرجات أم لا؟

الشكل العام:

ISPRINTER(<*expC*>)

الشـــرح:

إذا كانت الطابعة متصلة بالحاسب وفي وضع ON فإن هذه الوظيفة ستشتمل على القيمة المنطقية .T.وإلا فستشتمل على .F.

الاختلاف عن *dBASE III PLUS* : لا يوجد بها.

مثـــال:

البرنامج التالي يطبع تقرير على الطابعة إذا كانت جاهزة فإذا لم تكن جاهزة فإنه يظهر رسالة تتيح إما إلغاء العمل بالضغط على مفتاح ESC أو إعادة المحاولة بضغط أي مفتاح آخر.

```
DO WHILE INKEY # 27
   IF .NOT. ISPRINTER        && إذا لم تكن الطابعة جاهزة
     WAIT "Printer not ready...press Esc to cancel" + ;
      " any other key to retry"
     INKEY(0)
     LOOP
   ELSE
     CLEAR
     @ 1,2 SAY "Printing report.... plrase wait"
     SET CONSOLE OFF
     SET DEVICE TO PRINTER
     USE STOCK INDEX ITEM_NO
     REPORT FORM STKRPT
   ENDIF
ENDDO
```

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

SET DEVICE – SET PRINT

الوظيفة ISUPPER( )

تحدد هل تبدأ العبارة بحرف كبير أم لا .

الشكل العام:

ISUPPER<*expC*>)

حيــــث :

<*expC*> : عبارة أو حقل حرفي .

الشـــــرح :

تستخـدم هذه الـوظيفة لمعرفة ما إذا كانت عبارة أو حقل حرفي تبدأ بأحد الحروف الكبيرة (Upper Case Letter) أو لا . فإذا كانت الاجابة صحيحة فستعطيك .T. بمعنى صح وإلا .F. بمعنى خطأ .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثــــال :

```
ALPH1="MAGDI"
? ISUPPER(ALPH1)                && .T. الاجابة
```

المكتبة : EXTEND.LIB

الأوامر ذات الصلة :

ISALPHA( ) – ISLOWER( ) – LOWER( ) – UPPER( )

الوظيفة LASTKEY( )

تعطي الرقم الشفرة الأمريكية (ASCII) المقابل لآخر حرف تم الضغط عليه أثناء توقف البرنامج مؤقتا.

الشكل العام:

LASTKEY( )

الشـــرح:

إذا توقف البرنامج نتيجة لأمر READ فإنه يقبل بيانات من المستخدم قبل استئناف العمل ويتم تخزين الشفرة المقابلة لآخر حرف اختير من لوحة المفاتيح في هذه الوظيفة. فإذا لم تضغط أي مفتاح فستشتمل على الرقم صفر. ويمكن استخدام هذه الوظيفة للتفريع في البرنامج بناء على آخر مفتاح يتم الضغط عليه كما سيتضح من المثال التالي.

الاختلاف عن ***dBASE III PLUS*** : لا توجد بها.

مثـــال:

البـرنامج التالي برنامج مبسط لادخال بيانات إلى ملف STOCK.DBF وهو يسمح لمدخل البيانات بإلغاء المدخلات إذا ضغط على مفتاح ESC . وتقوم الوظيفة LASTKEY( ) بمراجعة آخر مفتاح تم الضغط عليه. فما لم يكن مفتاح Esc تقوم كلبر بإضافة سجل خال في نهاية الملف واستبدال محتوياته بالمدخلات.

```
USE STOCK
STORE SPACE(5) TO M_ITEM
STORE SPACE(20) TO M_DESC
STORE 0 TO M_QTY
CLEAR
@ 5,5 SAY "Enter item No....: " GET M_ITEM
@ 7,5 SAY "      description: " GET M_DESC
@ 9,5 SAY "    r quantity...: " GET M_QTY
READ
IF LASTKEY() = 27   && Esc اذا ضغط مفتاح
   CLEAR
   RETURN
ENDIF
APPEND BLANK
REPLACE ITEM_NO WITH M_ITEM, DESC WITH M_DESC, QTY WITH M_QTY
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

INKEY( )

الوظيفة LASTREC( )/RECCOUNT( )

تعطي إجمالي عدد سجلات الملف المفتوح.

الشكل العام:

LASTREC( ) أو RECCOUNT( )

الشـــــرح:

تستخـدم هذه الوظيفة لمعرفة عدد سجلات الملف المفتوح وهي تعطي عدد سجـلات الملف بصفـة مطلقـة أي بصرف النـظر عن تأثـير بعض الأوامـر مثـل SET FILTER أو SET DELETED ON .

ويمـكـن اسـتخــدام هذه الــوظيفــة مع كل من وظيفــة RECSIZE( ) و DISKSPACE( ) لمعـرفة الحجم الحقيقي للملف داخل البرنامج وتحديد المساحة المتبقية على القرص إذا كنت ستعمل نسخا احتياطية من البرنامج للملف مباشرة.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

لمعرفة عدد سجلات ملف STUDENTS.dbf .

```
USE STUDENTS
? RECCOUNT()
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

RECNO( ) – RECSIZE( ) – DISPSPACE( ) – DBF( )

الوظيفة LEFT()

تستخرج عددا من الحروف ابتداء من يسار العبارة الحرفية.

الشكل العام:

LEFT(<*expC*>, <*expN*>)

حيـــث:

<*expC*> : عبارة أو حقل حرفي.

<*expC*> : رقم أو تعبير رقمي.

الشـــرح:

تستخـدم هذه الوظيفة لاستخراج عدد من الحروف (<*expN*>) ابتداءً من يسار عبارة أو حقل حرفي (<*expC*>) : وهذه الوظيفة مشابهة للوظيفة SUBSTR( ) بفرض أن SUBSTR( ) تبدأ من العمود الأول. إذا حددت عددا من الحروف تزيد عن عدد الحروف التي تشتمل عليها العبارة أو الحقل الحرفي فإن الوظيفة ستستخرج كل العبارة أو الحقل وبالطبع إذا كان عدد الحروف صفرا أو رقما سالبا فلن تستخرج لك شيء.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

```
MNAME = "Abu Al-Ata"
? LEFT(MNAME,3)              && Abu  الاجابة
```

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

AT( ) – ALLTRIM( ) – RIGHT( ) – STUFF( ) – SUBSTR( )

الوظيفة LEN( )

تعطي طول عبارة حرفية.

الشكل العام:

LEN(<*expC*>/<*array*>)

حيــــث:

<*expC*>/<*array*> : عبارة أو حقل حرفي.

الشـــــرح:

تستخدم هذه الوظيفة لمعرفة عدد الحروف التي تشتمل عليها عبارة أو حقل حرفي أو مصفوفة فإذا كانت العبارة أو الحقل الذي تسأل عنه لا يشتمل على شيء فسيكون الجواب صفرا.

ويمكن استخدام هذه الوظيفة داخل البرنامج لاتخاذ قرار بناء على ما إذا كانت العبارة أو الحقل يحتوي على حرف أم لا. لتوضيح ذلك انظر المثال التالي.

الاختــلاف عن *dBASE III PLUS* : لا تتعـامـل «دي بيس ثري بلاس» مع المصفوفات، وتتعامل مع العبارة الحرفية.

مثـــال:

المثال الآتي جزء من برنامج يستخدم لادخال فواتير البيع في شركة معدات أدوات صنـاعية فإذا انتهى المستخدم من إدخال الفواتير فيجب أن يضغط مفتاح الادخال للخروج.

وفي هذا المثـال استخـدمت الـوظيفة TRIM( ) مع الوظيفة LEN( ) لحذف المسافات الموجودة على يمين العبارة وبما أن العبارة تشتمل على فراغات فقط فبعد حذف جميع الفراغات لن تحتوي على شيء أي سيصير طولها 0 .

```
MINVOICE = SAPCE(10)
@ 24,12 SAY ;
"Enter invoice no., or press Enter to quit" GET MINVOICE
RESD
IF LEN(TRIM(MINVOICE)) = 0
   CLEAR
   RETURN
ENDIF
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

TRIM( ) – STR( ) – SUBSTR( )

الوظيفة LOG()

تعطى اللوغاريتم الطبيعي لأي رقم.

الشكل العام:

LOG(<*expN*>)

حيــث:

<*expN*> : أي رقم أو تعبير رقمي.

الشـــرح:

تعطى هذه الوظيفة اللوغاريتم الطبيعي لأي رقم أو تعبير رقمي. وعادة تبدأ لوغارتمات الأرقام بالقيمة الثابتة 2.7182818285

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثـــال:

للحصول على اللوغاريتم الطبيعي للرقم 2.7182818285

```
? LOG(2.7182818285)          && 1.000000000  الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET DECIMAL – SET FIXED – EXP( )

الوظيفة LOWER( )

تحول الحروف الكبيرة (Upper Case) إلى صغيرة (Lower Case) .

الشكل العام:

LOWER(<expC>)

حيــث:

<expC> : حقل أو عبارة حرفية.

الشــرح:

تقوم هذه الوظيفة بتحويل الحروف الانجليزية الكبيرة (Upper Case Letters) إلى حروف صغيرة (Lower Case Letters) .

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثــال:

المثال التالي يستخدم هذه الوظيفة:

```
STORE "GOOD MORNING" TO MDAY
? LOWER(MDAY)              &&  good morning  الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

UPPER( ) – ISLOWER( ) – ISUPPER( )

الوظيفة LTRIM()

تحذف المسافات الخالية الموجودة على يسار عبارة حرفية.

الشكل العام:

LTRIM(<*expC*>)

حيـــث:

<*expC*> : حقل أو عبارة حرفية.

الشـــرح:

تستخدم هذه الوظيفة لحذف المسافات الخالية الموجودة على يسار عبارة حرفية. وتستخدم بصفة خاصة لحذف المسافات التي تنشأ نتيجة استخدام الوظيفة STR( ) .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

يوضح لك المثال الآتي كيف تتخلص من المسافات الزائدة والموجودة على يسار الرقم 35 الناتجة من استخدام وظيفة STR( ) .

```
AGE = 39
? "I am " + STR(AGE) + "years old"
** Clipper gives next answer
** I am            35 years old
? "I am " + LTRIM(STR(AGE)) + "years old"
** Clipper gives next answer
** I am 35 years old
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

STR( ) – TRIM( ) – RTRIM( ) – LEFT( ) – RIGHT( ) – SUBSTR( )

الوظيفة LUPDATE()

تعطي تاريخ آخر تعديل في الملف.

الشكل العام:

LUPDATE( )

الشـــرح:

تتيح لك هذه الوظيفة معرفة تاريخ آخر تعديل تم على الملف المفتوح. ويظهر التاريخ بالشكل الأمريكي (mm/dd/yy) ما لم تغيره بأمر SET DATE أو تستخدم أمر SET CENTURY وتفيد هذه الـوظيفة في حالة التعديلات التي يجب أن تتم على الملفات لمرة واحدة فقط في اليوم.

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثـــال:

المثـال الآتي جزء من برنامج يستخدم لعمل نسخ احتياطية (Back up) من ملف STOCK.dbf لمرة واحدة فقط في نهاية اليوم.

```
USE STOCK
IF LUPDATE() # DATE()
   @ 24,1 SAY "Last back up date: " + DTOC(LUPDATE())
   * Back up commands
ENDIF
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

DBF( ) – RECCOUNT( ) – RECSIZE( )

الوظيفة MAX()

تعطى القيمة العظمى من قيمتين.

الشكل العام:

MAX(<*expD1*> / <*expN1*>, <*expD2*> / <*expN2*>)

حيـــث:

<*expN1*> أو <*expN2*> : رقم أو تعبير رقمي.

<*expD1*> أو <*expD2*> : تاريخ أو عبارة تاريخية.

الشـــرح:

تحدد هذه الوظيفة القيمة العظمى من قيمتين رقميتين أو بين تاريخين وتفيد هذه الوظيفة في إظهار عمر أصغر طالب أو إظهار حد أدنى لأقل سعر أو أقل راتب، كما سيتضح من المثال التالي.

الاختلاف عن *dBASE III PLUS* : «دي بيس ثري بلاس» لا تقارن بين تاريخين.

مثـــال:

المثـال التـالي يستخدم ملف STOCK.dbf لاظهار أكبر القيمتين: السعر أم الرقم ١٠٠ بالاضافة إلى السعر المسجل بالملف.

```
USE STOCK
LIST PRICE,MAX(100,PRICE)
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

MIN( )

الوظيفة MEMOEDIT()

تفتح نافذة تشبه أمر BROWS في «دي بيس» لتعديل حقل الملاحظات أو عبارة حرفية.

الشكل العام:

MEMOEDIT(<*expC1*>, [<*expN1*>] [,<*expN2*>] [,<*expN3*>] [,<*expN4*>] [,<*expL1*>] [,<*expC2*>] [,<*expN5*>] [,<*expN6*>] [,<*expN7*>] [,<*expN8*>] [,<*expN9*>] [,<*expN10*>]

حيث:

<*expC1*> : اسم حقل الملاحظات (Memofield) أو العبارة الحرفية.

ملاحظة: يمكن حسب اختيارك إضافة بعض أو كل الاختيارات التالية

<*expN1*> <*expN2*> <*expN3*> <*expN4*>

حدود النافذة على الشاشة بهذا الترتيب: السطر الأول ـ العمود الأيسر ـ السطر الأخير ـ العمود الأيمن. . فإذا لم تختر حدوداً للنافذة فستظهر على الشاشة كلها.

<*expL1*> : تحدد هل بإمكان المستفيد تعديل حقل الملاحظات أم الاطلاع عليه فقط؟ فإذا كانت .T. فيمكن للمستفيد تعديل محتويات الحقل أو العبارة. أما إذا كانت .F. فسيقتصر الأمر على رؤية البيانات.

<*expC2*> : اسم وظيفة خاصة يتم تنفيذها عند ضغط مفتاح غير تلك المفاتيح المعرفة للوظيفة MEMOEDIT( )

<*expN5*> : رقم يحدد طول السطر داخل النافذة. إذا زاد طول السطر عن عرض النافذة فيستكمل السطر دائما في السطر التالي له.

<*expN6*> : رقم يحدد الفراغات التي ستقفز عندما يتم ضغط مفتاح Tab

<expN7> : أول سطر من حقل الملاحظات سيظهر داخل النافذة.

<expN8> : أول عمود من حقل الملاحظات سيظهر داخل النافذة.

<expN9> : مكان أول سطر سيظهر داخل النافذة (إذا لم يحدد فيعتبر صفرا).

<expN10> : مكان أول عمود سيظهر داخل النافذة (إذا لم يحدد فيعتبر صفرا).

الشـــرح:

هذه الوظيفة تقوم بعمل أمر BROWSE في «دي بيس ثري بلاس». فهي تسمح بإظهار نافذة على الشاشة تشتمل على محتويات حقل ملاحظات (Memofield) أو عبارة حرفية. وتسمح كذلك بتحريك المؤشر داخل هذه النافذة لاجراء التعديلات المطلوبة ويمكن التحكم في أول سطر سيظهر على الشاشة وفي مكان ظهور المؤشر بإضافة اختيارات معينة إلى الشاشة.

وعند ظهور النافذة التي تشتمل على بيانات الحقل المطلوب يمكنك التحرك باستخـدام مفـاتيح الأسهم ومفـاتيح أخرى. ويوضح الجدول التالي المفاتيح التي تستخدم مع نافذة MEMOEDIT( ) ووظيفة كل منها:

المفتاح	وظيفتـــه
↑ ↓ ←→	تسمح بتحريك المؤشر في اتجاه السهم.
Ctrl-A	تحريك المؤشر كلمة لجهة اليسار.
Ctrl-F	تحريك المؤشرة كلمة لجهة اليمين.
Home	أول السطر.
End	آخر السطر.
Ctrl-Home	أول سطر في النافذة.
Ctrl-End	آخر سطر في النافذة.

المفتاح	وظيفتـــه
PgUp	صفحة لأعلى.
PgDn	صفحة لأسفل.
Ctrl-PgUp	بداية حقل الملاحظات.
Ctrl-PgDn	آخر حقل الملاحظات.
Ctrl-Y	حذف سطر.
Ctrl-T	حذف كلمة.
Ctrl-W	الخروج مع حفظ التعديلات.
Esc	الخروج بدون حفظ التعديلات.

من الممكن أن تشتمل الوظيفة على وظيفة خاصة (<*expC2*>) لتقوم بالتحكم في وظائف تعديل أو الاطلاع على حقل الملاحظات أو العبارة الحرفية.

الاختلاف عن dBASE III PLUS : لا توجد بها.

مثـــال:

يشتمل ملف الطلاب على حقل ملاحظات (Memo field) اسمه NOTES ، فإذا أردت إظهار محتويات هذا الحقل بغرض تعديله. فيمكنك استخدام المثال التالي لهذا الغرض. والمثال يفترض أن التعديل يتم بناء على رغبة المستفيد إذا اختار Y رداً على رسالة معينة.

```
USE STUDENTS
YN = " "
@ 24,5 SAY "Edit memo field? [Y/N]" GET YN
  IF UPPER(YN) = "Y"
     SAVE SCREEN
     REPLACE NOTES WITH MEMOEDIT(NOTES,05,05,20,75,.T.)
  ENDIF
RESTORE SCREEN
```

وهذا الجزء من البرنامج يضع محتويات الشاشة بالذاكرة إذا اختار المستفيد "Y" رداً على الرسالة بمعنى نعم أريد التعديل،بعد ذلك تظهر نافذة ابتداء من السطر الخامس والعمود الخامس وانتهاء بالسطر العشرين والعمود الخامس والسبعين لتعديل محتويات حقل NOTES . وسيتم حفظ التعديلات لأن النافذة ستظهر مع أمر REPLACE . وبعد الانتهاء من تعديل محتويات الحقل داخل النافذة، ستظهر الشاشة الأولى التي كانت موجودة قبل تعديل حقل الملاحظات.

فإذا أردت أن تظهر النافذة السابقة وتظهر محتويات حقل الملاحظات لغرض فحصها فقط بدون صلاحيات التعديل استخدم المثال بهذا الشكل.

```
USE STUDENTS
YN = " "
@ 24,5 SAY "Edit memo field? [Y/N]" GET YN
  IF UPPER(YN) = "Y"
    SAVE SCREEN
    REPLACE NOTES WITH MEMOEDIT(NOTES,05,05,20,75,.F.)
  ENDIF
RESTORE SCREEN
```

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

DBEDIT( ) – MEMOLINE( ) – MEMOREAD( ) – MEMOTRAN( ) – MEMOWRIT( ) – MLCOUNT( )

الوظيفة MEMOLINE()

تسمح بإظهار (أو طباعة) سطر موجود في عبارة حرفية أو حقل ملاحظات.

الشكل العام:

MEMOLINE (<*expC*>, <*expN1*>, <*expN2*>)

حيـــث:

<*expC*> : اسم العبارة أو الحقل.

<*expN1*> : عرض السطر المطلوب إظهاره.

<*expN2*> : رقم السطر المطلوب إظهاره.

الشـــرح:

تستخدم هذه الوظيفة لاظهار محتويات حقل ملاحظات (Memofield) أو عبارة حرفية بشكل مختلف يسهل قراءته عما هي عليه. ولكي تحسب عدد السطور الموجودة داخل العبارة أو الحقل استخدم الوظيفة MLCOUNT( ) التي تحسب عدد السطور التي يشتمل عليها الحقل أو العبارة ثم استخدم هذه الوظيفة لاظهار السطر/ السطور التي تريد إظهارها. إذا كان رقم السطر المحدد في الوظيفة MEMOLIN( ) غير موجود في العبارة أو الحقل فلن يُستخرج شيء.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

يستخدم ملف الطلاب (Students) حقل ملاحظات (Memo) اسمه NOTES لتسجيل معلومات مختلفة عن الطالب..

المثال التالي تستخدم الوظيفة MEMOLINE( ) لطباعة حقل الملاحظات NOTES . بشكـل منـظم على هيئة تقرير يشتمل السطر الواحد على ٥٥ حرفاً.

ويستخدم أيضا الوظيفة COUNT( ) لحساب عدد السطور التي يشتمل عليها حقل الملاحظات.

```
USE STUDENTS
SET DEVICE TO PRINT
LINES_NO = MLCOUNT(NOTES,55)    && احسب عدد سطور حقل الملاحظات NOTES
                                && بفرض أن عرض السطر 55 حرفا
FOR M_LINE = 1 TO LINES_NO      && دوارة بعدد السطور
  PRTLINE = MEMOLINE(NOTES,55,M_LINE)
  IF PRTLINE == " "             && اذا كان حقل PRTLINE
                                && لايشتمل على شيء
    EXIT
    EJECT
  ENDIF
  @ PROW()+1,2 SAY PRTLINE
NEXT
SET DEVICE TO SCREEN
```

يشتمل البرنامج على جملة IF ومعناها إذا صارت محتويات PRTLINE لاشيء أوقف الطباعة وانقل الطابعة إلى بداية الصفحة التالية، وتصبح محتويات PRTLINE لاشيء إذا انتهت سطور حقل الملاحظات (أي إذا أشارت M LINE إلى رقم سطر غير موجود بحقل الملاحظات).

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

MEMOEDIT( ) – MLCOUNT( ) – HARDCR( )

الوظيفة MEMOREAD()

تقرأ ملف نصي (ASCII) من القرص الممغنط.

الشكل العام:

MEMOREAD (<expC>)

حيث:

<expC> : اسم الملف المطلوب قراءته.

الشرح:

تقرأ هذه الوظيفة ملف نصي مكتوب بشفرة ASCII موجود على القرص. وبالتالي يمكنك نقله إلى الذاكرة داخل حقل ذاكرة بشرط ألا يزيد حجمه عن ٣٢ك.ب. أو إلى حقل ملاحظات (MEMO) بشرط ألا يزيد حجمه عن ٦٤ك.ب. أو تعديل محتوياته بالوظيفة MEMOEDIT() . ويجب ذكر الاسم الممتد مع اسم الملف المطلوب.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثال:

المثال التالي يستخدم الوظيفة MEMOREAD( ) لنقل ملف COURSES.TXT إلى حقل الملاحظات NOTES الموجود في ملف STUDENTS.DBF

```
USE STUDENTS
REPLACE NOTES WITH MEMOREAD("COURSES.TXT")
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

MEMOEDIT( ) – MEMOWRIT( )

الوظيفة MEMORY()

تحسب المساحة المتبقية من الذاكرة.

الشكل العام:

MEMORY (0)

الشـــرح:

تحسب المساحة المتبقية من الذاكرة بالكيلوبايت وتفيد هذه الوظيفة في حالات كثيرة مثل الحاجة لمعرفة المساحة المتبقية من الذاكرة لتشغيل أحد أوامر DOS أو لمعرفة المساحة التي تتطلبها عملية ترجمة البرنامج .

الاختلاف عن dBASE III PLUS : لا توجد بها .

مثـــال:

المثال الآتي عبارة عن برنامج يحسب المساحة المتبقية من الذاكرة والمساحة التي يشغلها البرنامج ويظهر رسالتين بكل منهما .

```
MEM_AVA = MEMORY(0) * 1024          && لاظهار المساحة المتوفرة بالحروف
MEM_TOT = 655360                    && 640 * 1024
MEM_OCC = MEM_TOT - MEM_AVA         && المساحة المشغولة من الذاكرة
? "Available memry: " + STR(MEM_AVA,8,0)
? "Used memory: " + STR(MEM_OCC,8,0)
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة : لا توجـــد .

الوظيفة MEMOWRIT( )

تنقل محتويات حقل ملاحظات أو عبارة حرفية إلى ملف نصي على القرص المغنط.

الشكل العام:

MEMOWRIT (<*expC1*>, <*expC2*>)

حيـــث:

<*expC1*> : اسم الملف المطلوب كتابته (يشمل الاسم الممتد).

<*expC2*> : اسم الملف المطلوب نقل محتوياته.

الشـــرح:

بعـد نقـل محتـويات الحقـل الحـرفي أو العبـارة الحـرفية تشتمـل الـوظيفـة MEMOWRIT( ) على القيمة المنطقية .T. أما إذا لم تتم عملية النقل فستشتمل على القيمة المنطقية .F.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال التالي يكتب محتويات حقل NOTES الموجود في ملف STUDENTS.DBF إلى ملف نصي على الـقــرص باسم STUDENTS.DOC فإذا نجحت عملية الكتابة يُظهِر رسالة تفيد نجاح عملية الكتابة وإلا فيُظهِر رسالة تفيد أن عملية الكتابة لم تتم.

```
USE STUDENTS
FLAG = MEMOWRIT("STUDENT.DOC", NOTES)
IF FLAG
   ? "Write ok. "
ELSE
   ? "Unable to write"
ENDIF
```

المثـال التـالي يستخدم ثلاث وظائف لقراءة ملف نصي موجود على القرص وتعديله ثم كتابته على القرص مرة ثانية.

```
USE STUDENTS
MEMOWRIT("COURSES.DOC", MEMOEDIT(MEMOREAD("COURSES.DOC")))
```

المكتبة : **EXTEND.LIB**

الأوامر ذات الصلة :

MEMOREAD( ) – MEMOEDIT( )

الوظيفة MIN( )

تعطى القيمة الصغرى من قيمتين.

الشكل العام:

MIN(<*expD1*>/ <*expN1*>, <*expD2/expN2*>)

حيـــث:

تستبدل كل من <*expN1*> و <*expN2*> برقم أو تعبير رقمي. بينما كل من <*expD1*> و <*expD2*> بتاريخ أو عبارة تاريخية.

الشـــرح:

تحدد هذه الوظيفة القيمة الصغرى من قيمتين رقميتين. وتفيد هذه الوظيفة في تحديد حد أقصى لأعلى سعر أو أعلى راتب. . . الخ.

الاختلاف عن *dBASE III PLUS* : لا تقارن «دي بيس» بين تاريخين.

مثـــال:

يستخدم المثال التالي ملف STOCK.dbf لاظهار أصغر القيمتين: السعر أم الرقم ١٠٠ بالاضافة إلى السعر المسجل بالملف.

```
USE STOCK
LIST PRICE, MIN(100,PRICE)
```

بينما تستخدم المثال التالي لاظهار التاريخ الأصغر: تاريخ اليوم أو تاريخ الاستحقاق.

```
DUEDATE = CTOD("12/31/90")
? MIN(DATE(),DUEDATE
USE STUDENTS
STARTING = MLPOS(NOTES,55,3)
? SUBSTR(NOTES,STARTING,15)
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

MAX()

الوظيفة MLCOUNT( )

تحسب عدد السطور الموجودة في حقل الملاحظات أو في عبارة حرفية تبعا لعدد حروف السطر الواحد.

الشكل العام:

MLCOUNT (<*expC*> , <*expN*>)

حيث:

<*expC*> : اسم حقل الملاحظات أو العبارة الحرفية.

<*expN*> : عدد حروف كل سطر.

الشرح:

تحسب هذه الوظيفة عدد السطور الموجودة في حقل الملاحظات أو عبارة حرفية طبقا لعدد الحروف المحددة وتستخدم هذه الوظيفة بصفة أساسية مع الوظيفة MEMOLINE( ) التي تظهر أو تطبع سطور حقل الملاحظات أو العبارة الحرفية (راجع الوظيفة MEMOLINE( )).

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثال:

راجع المثال الموجود بالوظيفة MEMOLINE( ) .

المكتبة : EXTEND.LIB

الأوامر ذات الصلة:

MEMOLINE( )

الوظيفة MLPOS( )

تحدد موقع سطر معين داخل حقل ملاحظات أو تعبير حرفي.

الشكل العام:

MLPOS (<*expC*> , <*expN1*> , <*expN2*>)

حيـــث:

<*expC* > : اسم الحقل أو العبارة.

<*expN1*> : عدد الحروف في السطر الواحد الذي تستخدمه كلبر في حساب عدد السطور.

<*expN2*> : رقم السطر.

الشـــرح:

تحدد هذه الوظيفة موقع سطر معين من بداية حقل الملاحظات أو تعبير حرفي يشتمل على أكثر من سطر، ويتم حساب عدد السطور بناء على عدد حروف السطر التي تدخل مع الوظيفة (<*expN1*>) .

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال التالي يوضح استخدام هذه الوظيفة. ويشتمل على أمرين:

الأول يحسب أين يبدأ السطر الثالث داخل حقل الملاحظات NOTES الموجود بملف STUDENTS.DBF باعتبار أن طول السطر هو ٥٥ حرفاً. ويضع الناتج في حقل الذاكرة STARTING .

الثاني يظهر الحروف الخمسة عشر التي تبدأ من موقع السطر الثالث داخل حقل الملاحظات .

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

MEMOREAD( ) – MLCOUNT( )

الوظيفة MONTH( )

تظهر رقم الشهر المسجل بحقل أو عبارة تاريخية.

الشكل العام:

MONTH(<*expD*>)

حيـــث:

<*expD*> : حقل أو عبارة رقمية.

الشـــرح:

تظهر رقما يدل على ترتيب الشهر في السنة وبما أن عدد شهور السنة ١٢ شهرا فيجب أن يكون الرقم من ١ ـ ١٢. ويجب أن تكون محتويات <*expD*> حقلا تاريخيا أو عبارة تاريخية.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

المثال التالي يظهر رقم الشهر الموجود بالتاريخ المسجل بالحاسب

```
?MONTH (DATE( ))
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

DATE( ) – DAY( ) – YEAR( )

الوظيفة NETERR( )

تستخدم مع شبكة الاتصالات لتحدد هل نجح أمر USE أو أمر APPEND BLANK أم لا؟

الشكل العام:

NETERR( )

الشــــرح:

تشتمل هذه الوظيفة على القيمة المنطقية .T. بمعنى True (صح) إذا حدث خطأ أثناء فتح الملف أو استخدام أمر APPEND BLANK . ويحدث هذا الخطأ إذا حاولت فتح الملف بينما هو يستخدم استخداما منفردا بواسطة مستفيد آخر في الشبكة. وأيضا إذا حاولت استخدام أمر APPEND BLANK لاضافة سجل خال في نهاية الملف بينما يقوم مستفيد آخر بنفس المحاولة في نفس اللحظة فسيحدث نفس الخطأ.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال التالي يفتح ملف الفهرس إذا تم فتح ملف قاعدة البيانات بنجاح، وإلا يظهر رسالة تفيد عدم فتح الملف ويستخدم الوظيفة NETERR( ) للتعرف على حالة الملف هل هو مفتوح أم لا؟

```
USE STOCK
IF .NOT. NETERR()
   SET INDEX TO ITEM_NO
ELSE
   ? " Cann't open the file... Network error"
   ? " Press any key to return to previous menu"
   WAIT ""
   CLEAR ALL
   RETURN
ENDIF
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

USE – FLOCK( ) – SET EXCLUSIVE

الوظيفة NETNAME()

تعطي اسم المحطة المتصلة بشبكة الاتصالات المستخدمة.

الشكل العام:

NETNAME()

الشـــرح:

تستخـدم هذه الـوظيفـة بصفة خاصة لمعرفة اسم المحطة المفتوحة في حالة التطبيقات التي تستخدم شبكة الاتصالات المحلية. فإذا لم تكن المحطة معرفة لشبكة الاتصالات فلن يظهر شيء.

الاختلاف عن dBASE III PLUS : لا توجد بها.

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة: لا توجــد.

الوظيفة NEXTKEY( )

تعطي الشفرة المقابلة لأول حرف موجود في المحطة الانتقالية للوحة المفاتيح (Keyboard buffer) دون أن تمسحه.

الشكل العام:

NEXTKEY( )

الشــــرح:

تعطي هذه الوظيفة الشفرة الأمريكية (ASCII) المقابلة لأول حرف موجود في المحـطة الانتقالية للوحة المفاتيح. وبينما تمسح الوظيفة NKEY( ) هذا الحرف من المحطة الانتقالية، فإن هذه الوظيفة تحتفظ به. والميزة من ذلك هي إمكانية استخدام الحرف الذي أدخل من لوحة المفاتيح للتفريغ أو لتنفيذ برنامج آخر. والشفرة التي تعـطيها هذه الوظيفة للحروف ومفاتيح الوظائف ومفاتيح التحكم... الخ. هي نفس الشفرة التي تعطيها الوظيفة INKEY( ) والوظيفة LASTKEY( ).

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال التالي يضع الكود المقابل لحرف Ctrl-End في المحطة الانتقالية للوحة المفاتيح (Keyboard buffer) ثم يسأل عن قيمة الوظائف الثلاث.. لاحظ الفرق في النتائج.

```
KEYBOARD CHR(23)                        && CHR(23) = Ctrl-End
? NEXTKEY(), INKEY(), LASTKEY()         && 23  27  27  الاجابة
```

المكتبة : EXTEND.LIB

الأوامر ذات الصلة:

INKEY( ) – LASTKEY( )

الوظيفة PCOL()

تعطي مكان العمود الذي ستبدأ الطابعة الطباعة ابتداء منه.

الشكل العام:

PCOL()

الشـــرح:

تستخدم هذه الوظيفة داخل البرنامج في الغالب لتوجيه الطابعة لتبدأ الطابعة من مكان محدد. ويمكن أن تستخدم لتحديد مكان جديد للطباعة بإضافة عددا من الأعمدة إلى المكان الذي تقف عنده الطابعة. ويمكنك تخزين قيمة PCOL() بالذاكرة.

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثـــال:

لتوجيه الطابعة لتبدأ الطباعة بعد عشرة أعمدة من المكان الذي تقف عنده استخدم الأوامر التالية:

```
SET DEVICE TO PRINT
@ 1, PCOL() + 10 SAY "Is it clear? "
SET DEVICE TO SCREEN
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@ – PROW() – ROW() – COL()

الوظيفة PCOUNT( )

تحسب عدد المعطيات (Parameters) التي دخلت إلى البرنامج من برنامج آخر أو من لوحة المفاتيح .

الشكل العام:

PCOUNT( )

الشــــرح:

تعــطي هذه الـوظيفـة عدد المعـطيات (Parameters) التي قُبلت من خارج البرنامج الذي ينفذ أو الوظيفة الخاصة. فإذا لم تقبل معطيات من خارج البرنامج أو الوظيفة الخاصة فستشتمل الوظيفة على الرقم صفر.

الاختلاف عن ***dBASE III PLUS*** : لا توجد بها.

مثــــال:

المثال التالي يسأل عن اسم العميل ليبدأ البحث عنه داخل ملف CUSTOMER.DBF ويقبل اسم العميل من خارج البرنامج - من لوحة المفاتيح - فإذا لم يُدخل المشغل اسم العميل فإنه يطلب منه إدخال اسم العميل.

```
* Search.prg
USE CUSTOMER INDEX NAME
PARAMETERS M_NAME
IF PCOUNT() = 0
   ACCPET "Enter customer name: " TO M_NAME
ENDIF
SEEK M_NAME
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

DO...WITH – PARAMETERS

الوظيفة PROCNAME( ) - PROCLINE( )

تحدد PROCNAME( ) اسم البرنامج أو الاجراء الذي ينفذ. وتحدد PROCLINE( ) رقم السطر الحالي.

الشكل العام:

PROCLINE( )

PROCNAME

الشـــرح:

تعطي الوظيفة PROCLINE( ) رقم السطر الحالي داخل البرنامج أو الاجراء الذي ينفذ وينسب هذا الرقم إلى بداية البرنامج. إذا تمت ترجمة البرنامج بدون ترقيم الأوامر - إذا اخترت I - أثناء الترجمة - فإن هذه الوظيفة لن تستطيع التعرف على رقم السطر. بينما تعطي الوظيفة PROCNAME( ) اسم البرنامج أو الاجراء الذي ينفذ حاليا. وتستخدم هاتان الوظيفتان لاظهار معلومات عن البرنامج أو الاجراء الذي ينفذ أثناء التنفيذ.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال التالي يظهر اسم الاجراء SRCHPROC ورقم أول سطر موجود به عندما يبدأ تنفيذه.

```
DO SRCHPROC
? PROCNAME()+STR(PROCLINE(),3,0)        && SRCHPROC   1   الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة: لا توجـد.

الوظيفة PROW()

تحدد مكان السطر الذي ستبدأ منه الطابعة.

الشكل العام:

PROW()

الشـــــرح:

تستخدم هذه الوظيفة غالبا داخل البرنامج لتوجيه الطابعة لتبدأ الطابعة من سطر معين. ويمكن أن تستخدم لتحديد سطر جديد لبداية الطباعة بإضافة عدد من السطور إلى السطر الذي تقف عنده الطابعة. وغني عن البيان أنه لا يجوز طرح رقم من PROW() لأن ورق الطابعة لا يتحرك للخلف.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

إذا أردت أن تنتقل الطابعة إلى صفحة جديدة عندما تصل إلى السطر الخمسين أثناء طباعة بيانات الموردين استخدم هذا البرنامج.

```
USE VENDOR
DO WHILE .NOT. EOF()
   ? TRIM(NAME) + " " + TRIM(ADDRESS) + "," + TRIM(CITY) + "."
   IF PROW() > 50
      EJECT
   ENDIF
   SKIP
ENDDO
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@ – ROW( ) – COL( ) – PCOL( )

الوظيفة RAT()

تبحث عن آخر حرف أو مجموعة حروف موجودة داخل عبارة حرفية.

الشكل العام:

RAT(<expC1> , <expC2>)

حيث:

<expC1> : الحرف أو العبارة المطلوب البحث عنها.

<expC2> : العبارة المطلوب البحث فيها.

الشرح:

تعمل هذه الوظيفة عكس وظيفة AT() التي تستخدمها «دي بيس ثري بلاس» أو «كلبر» أي أنها تبحث عن وجود حرف أو عبارة داخل عبارة أخرى ابتداءً من يمين العبارة. فإذا لم تجد «كلبر» العبارة التي تبحث عنها (<expC1>) داخل العبارة التي تبحث فيها (<expC2>) فإنها تشتمل على الرقم صفر.

الاختلاف عن dBASE III PLUS : لا توجد بها.

مثال:

المثال التالي يبحث في العبارة الرئيسية عن مكان العبارة الصغيرة مرة من اليسار إلى اليمين باستخدام RAT() ومرة أخرى من اليمين إلى اليسار باستخدام AT() .

```
STRNG1 = "my'
STRNG2 = "I find my book on my table"
? RAT(STRNG1,STRNG2)            && 19  الاجابة
? AT(STRNG1,STRNG2)             &&  8  الاجابة
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

AT() – SUBSTR()

الوظيفة READEXIT()

تسمح لمفتاحي الأسهم ↓ أو ↑ بإلغاء أثر أمر READ أو تمنع ذلك.

الشكل العام:

READEXIT([<*expL*>])

حيـــث:

<*expL*> : تعبير منطقي .T. أو .F.

الشـــرح:

استبدل <*expL*> بالقيمة .T. إذا أردت السماح لمفتاحي الأسهم ↑ ↓ بإنهاء أمر READ أو .F. إذا أردت إلغاء ذلك. والوضع التلقائي هو .F.

تنفيذ هذه الوظيفة بدون اختيارات معها يحول إلى عكس الوضع الذي هي عليه. فإذا كانت تشتمل على .T. صارت تشتمل على .F. والعكس صحيح.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثال التالي يسمح بإلغاء أمر READ باستخدام الأسهم ↑ أو ↓ ثم يعيد الوضع إلى ما كان عليه بعد الانتهاء من أمر READ.

```
STORE "N" TO YESNO
READEXIT(.T.)        && READ اسمح للسهم العلوى والسفلى بانهاء امر
@ 24,2 SAY "More records? [Y/N]" GET YESNO PICT "!"
READ
IF YESNO = "Y"
   LOOP
ENDIF
READEXIT()               && ضع الوظيفة فى عكس الوضع الذى هى عليه
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

@...SAY...GET – READINSERT( )

الوظيفة READINSERT()

تسمح بحالة إدخال حروف بين حروف موجودة وتسمى Insert mode أو تلغي ذلك.

الشكل العام:

READINSERT([<*expL*>])

حيـــث:

<*expL*> : تعبير منطقي .T. أو .F.

الشـــرح:

إذا أردت إدخال حروف بين حروف موجودة أثناء توقف البرنامج نتيجة لأمر READ استبدل <*expL*> بالقيمة .T. ولالغاء هذه الامكانية استبدلها بالقيمة .F.

تنفيذ هذه الوظيفة بدون اختيارات مهما يحوِّل إلى عكس الوضع الذي هي عليه فإذا كانت تشتمل على .T. صارت .F. والعكس صحيح.

الاختلاف عن *dBASE III PLUS*: لا توجد بها.

مثـــال:

المثال التالي يسمح بإدخال حروف بين حروف العبارة STRNG حسب رغبة المستفيد إذا أراد إكمال اسم الأب.

```
READINSERT(.T.)                         && insert حول الى وضع
STRNG = "Abdullah M  Mohammad"
@ 24,2 SAY "Insert requested letters" GET STRNG
READ
READINSERT()                && ضع الوظيفة فى عكس الوضع الذى هى عليه
```

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

READ – READEXIT( )

الوظيفة READVAR()

تحدد اسم الحقل الذي يستدعى بأمر GET أو MENU .

الشكل العام:

READVAR( )

الشـــرح:

لأن هذه الوظيفة تعطي اسم الحقل الذي تم استدعاؤه بأمر GET أو MENU فيمكن استخدامها لاستدعاء شاشة معلومات مساعدة بناء على استدعاء حقل معين كما يمكن استخـدامها للتفريع داخل البرنامج أو تنفيذ إجراء معين بعد استدعاء حقل معين بأمر GET . إذا لم يستدع حقل أو متغير بأمر GET فلن تشتمل الوظيفة READVAR( ) على شيء.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

يقوم البرنامج التالي بتعديل حقول ملف STUDENTS.DBF . ويسمح بتنفيذ الاجراء SAYGET إذا ضغط المستفيد مفتاح F10 . وهذا الاجراء يظهر اسم الحقل الذي يجري تعديله في السطر الأخير من الشاشة.

```
USE STUDENTS
MFIRST   = SPACE(12)
MLAST    = SPACE(12)
MADDRESS = SAPCE(20)
MCITY    = SPACE(10)
STE KEY -9 TO SAYGET        && F10 نفذ SAYGET اذا ضغط مفتاح
@ 1,2 SAY "First name....." GET MFIRST
@ 3,2 SAY "Last name......" GET MLAST
```

```
@ 5,2 SAY "Address........" GET MADDRESS
@ 7,2 SAY "City .........." GET MCITY
READ
RETURN
*
PROCEDURE SAYGET
@ 24,2 SAY "Current field: " + READVAR()
RETURN
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

@...GET – MENU TO – READ – SET KEY

الوظيفة RECNO( )

تعطي رقم السجل الذي يقف عنده المؤشر.

الشكل العام:

RECNO( )

الشــــرح:

تمكنـك هذه الـوظيفة من معرفة رقم السجل الموجود تحت المؤشر في الملف المفتوح فإذا كان الملف لا يشتمل على سجلات فستعطيك الرقم ١ وستشتمل كل من الوظيفة EOF( ) والوظيفة BOF( ) على القيمة .T. .

إذا كان المؤشر يقف عند السجل رقم ١ وتم تحريكه للخلف سجلا واحدا فستظل هذه الوظيفة تعطي الرقم ١ أما إذا كان المؤشر عند آخر سجل وتم تحريكه للأمام سجلا إضافيا فستعطي الوظيفة رقم آخر سجل + ١ . وستشتمل الوظيفة EOF( ) على القيمة .T.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

يستخدم المثال التالي ملف STUDENTS.DBF وهو يشتمل على ١٥ سجلاً.

```
USE STUDENTS
? RECNO()                 && 1  الاجابة
GO BOTTOM
? RECNO()                 && 15  الاجابة
SKIP
? EOF()                   && .T.  الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

RECCOUNT( ) – LASTREC( )

الوظيفة RECSIZE( )

تعطي طول السجل داخل الملف.

الشكل العام:

RECSIZE( )

الشــــرح:

تعطي عدد الحروف التي يشتمل عليها كل سجل داخل الملف المفتوح، ويمكن استخدام هذه الوظيفة مع كل من وظيفة RECCOUNT( ) و DISKSPACE( ) لمعرفة الحجم الحقيقي للملف داخل البرنامج وتحديد المساحة المتبقية على القرص إذا كنت ستعمل نسخا احتياطية للملف من البرنامج مباشرة.

الاختلاف عن dBASE III PLUS: لا يوجد اختلاف.

مثــــال:

راجع الوظيفة DISKSPACE( ) والوظيفة HEADER( ).

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

RECOUNT( ) – DISKSPACE – HEADER( )

الوظيفة REPLICATE( )

تكرر حرف أو عبارة حرفية لعدد محدد من المرات.

الشكل العام:

REPLICATE(<*expC*>, <*expN*>)

حيـــث:

<*expC*> : الحرف أو العبارة المطلوب تكرارها.

<*expN*> : عدد المرات المطلوب تكرارها.

الشـــرح:

تستخدم هذه الوظيفة لتكرار حرف أو كلمة أو عبارة حرفية (<*expC*> لعدد من المرات مساويا للرقم المحدد بالعبارة <*expN*> .

ويجب ألا يزيد طول العبارة الناتج من استخدام الوظيفة عن ٣٢ك.ب للعبارة الحرفية أو ٦٤ك.ب لحقل الملاحظات.

ويمكن الاستفادة من هذه الوظيفة عندما ترغب في تكرار حرف أو جملة وكثيرا ما تحتاج لذلك عند رسم خطوط أو علامات أفقية.

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثـــال:

لتكرار العلامة "*" ٥٠ مرة:

```
?REPLICATE("*",50)
```

ولسماع صوت الجرس ٥ مرات متتالية:

```
?REPLICATE(CHR(7),5)
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SUBSTR( )

الوظيفة RESTSCREEN()

تستخرج جزء من شاشة حفظت بالوظيفة SAVESCREEN( )

الشكل العام:

RESTSCREEN(<*expN1*>, <*expN2*>, <*expN3*>, <*expN4*>, <*expC*>)

حيث:

<*expN1*> : رقم يحدد بداية الشاشة من اليسار.

<*expN2*> : رقم يحدد بداية الشاشة من اليمين.

<*expN3*> : رقم يحدد نهاية الشاشة من اليسار.

<*expN4*> : رقم يحدد نهاية الشاشة من اليمين.

<*expC* > : اسم حقل الذاكرة الذي يشتمل على الشاشة المحفوظة.

الشــرح:

تسمح الوظيفة SAVESCREEN( ) بحفظ جزء من الشاشة الحالية وتسمح الوظيفة RESTSCREEN( ) باسترجاع هذا الجزء. ويتم تخزين الشاشة بالألوان التي تبدو بها أمامك. وتختلف هاتان الوظيفتان عن أمري SAVE SCREEN و RESTORE SCREEN لأن أحدهما يقوم بحفظ شاشة كاملة والآخر يقوم باسترجاعها. ولذلك لا يصح استرجاع شاشة حفظت بأمر SAVE SCREEN بالوظيفة RESTSCREEN( ).

الاختلاف عن *dBASE III PLUS*: لا توجد بها.

مثـال:

في المثال التالي إذا ضغط المستفيد مفتاح F1 فسيتم استدعاء شاشة معلومات المساعدة في جزء من الشاشة وستظهر شاشة معلومات المساعدة في نفس الجزء الذي

تظهر عليه بيانات الموظف. ولذلك فإننا نحفظ هذا الجزء ثم نسترجعه بعد الانتهاء من قراءة معلومات المساعدة.

```
CLEAR
STE KEY -2 TO HELP          && HELP  اذا ضغط مفتاح F1 نفذ الاجراء
*
USE EMPLOYEE
STORE SPACE(10)  TO EMP_NO
STORE SPACE(20)  TO EMP_NAME
STORE 0          TO EMP_SAL,EMP_DEG
*
@ 5 ,2  SAY "Enter employee number:" GET EMP_NO
@ 6 ,2  SAY "Enter employee name  :" GET EMP_NAME
@ 7 ,2  SAY "Enter employee salary:" GET EMP_SAL
@ 8 ,2  SAY "Enter employee degree:" GET EMP_DEG
READ
*
PROCEDURE HELP
PARAMETERS X,Y,Z          && المعطيات تنتقل تلقائيا فى هذه الحالة
*
M_SAVE = SAVESCREEN(5,5,20,45)       && احفظ هذا الجزء من الشاشة
                                     && M_SAVE   فى حقل بالذاكرة اسمه
@ 5,5 TO 12,44 DOUBLE
@ 6,6 SAY "********** HELP MENU ********"
@ 6,6 SAY " Enter valid employee number"
* < more statements>
WAIT ""
*
RESTSCREEN(5,5,20,40,M_SAVE)
RETURN
```

المكتبة: **EXTEND.LIB**

الأوامر ذات الصلة:

SAVESCREEN( )

الوظيفة RIGHT()

تستنتج عددا من الحروف ابتداء من يمين العبارة الحرفية.

الشكل العام:

RIGHT(<expC>, <expN>)

حيـــث:

<expC> : عبارة أو حقل حرفي.

<expN> : رقم أو تعبير رقمي.

الشـــرح:

تستخدم هذه الوظيفة لاستخراج عدد من الحروف *<expN>* اعتبارا من يمين عبارة أو حقل حرفي *<expC>*.

إذا حددت عددا من الحـروف يزيد على عدد الحـروف التي تشتمل عليها العبارة أو الحقل الحرفي فإن الوظيفة ستستخرج كل العبارة أو الحقل وبالطبع إذا كان عدد الحروف *<expN>* صفرا أو رقما سالبا فلن ستخرج لك شي.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

المثـال التـالي يستخـرج الحروف الستــة الأولى من جهة اليمين من العبارة المخزونة في حقل الذاكرة MNAME

```
MNAME = "Magdi M. Abu Al-Ata"
? RIGHT(MNAME,6)              && Al-Ata  الإجابة
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

SUBSTR()

الوظيفة RLOCK( )/LOCK( )

تستخدم في شبكة الاتصالات المحلية لاغلاق سجل حتى لا يكتب عليه الآخرون.

الشكل العام:

RLOCK( ) أو LOCK( )

الشـــرح:

تستخدم كل من الوظيفة LOCK( ) أو RLOCK في حالة شبكة الاتصالات المحلية لنفس الغرض وهو إغلاق السجل حتى لا يستطيع أحد غير الذي أغلقه تعديل محتوياته، وإذا تم إغلاق السجل فإن الوظيفة ستشتمل على القيمة المنطقية .T. وإلا فستشتمل على .F. ويبقى السجل مغلقا حتى يفتحه الشخص الذي أغلقه بأمر UNLOCK أو يتم إغلاق الملف.

الاختـلاف عن dBASE III PLUS : في حالة «دي بيس ثري بلاس» تمنع هاتان الوظيفتان الآخرين من رؤية أو تعديل السجل. أما في «كلبر» فإنهما تمنعان الآخرين من تعديل السجل فقط.

مثـــال:

المثال التالي جزء من برنامج يستخدمه أكثر من بائع في شركة لبيع قطع غيار السيارات، فإذا فرض أن أحد البائعين يسأل عن صنف معين ليعرف الكمية الموجودة بالمخازن قبل بيعها، فلابد أن يغلق هذا السجل حتى لا يتمكن بائع آخر من تعديل نفس الكمية لنفس الصنف إلا بعد الانتهاء من تسجيل مبيعاته.

وفي هذا المثـال إذا لم يغلق السجـل قبل تسجيل عملية البيع وكانت الكمية الموجودة بالمخازن من الصنف هي ١٠ فإن كل بائع سيظهر أمامه نفس الكمية وبالتالي يحتمل أن يطلب أكثر من بائع بيع نفس الكمية.

```
USE STOCK INDEX  ITEM_NO
M_ITEM = SPACE(10)
@ 12,10 SAY "Enter item number: " GET M_ITEM
READ
SEEK M_ITEM
TRYING = 0
* The following loop to retry attempting locking if other user
* locking the same record
DO WHILE .NOT. LOCK() .AND. TRYING < 100
   TRYING = TRYING + 1
ENDDO
IF LOCK()            && اذا اغلق السجل بنجاح
  @ 2,5 SAY "Item number" + ITEM_NO
  @ 4,5 SAY "Item description" + DESC
  @ 6,5 SAY "Quantity        " GET QTY
  READ
ELSE
  @ 24,2 SAY "Sorry ... this item unavailable. Press any key"
  WAIT " "
  CLEAR
  CLEAR ALL
  RETURN
ENDIF
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET EXCLUSIVE – USE...EXCLUSIVE – FLOCK( ) – UNLOCK( )

الوظيفة ROUND( )

تقرب لأقرب رقم صحيح أو عشري.

الشكل العام:

ROUND(<*expN1*>, <*expN2*>)

حيـــث:

<*expN1*> : الرقم المطلوب تقريبه.

<*expN2*> : عدد الأرقام العشرية المطلوب التقريب إليها.

الشـــرح:

يقرب الرقم الموجود بالاختيار <*expN1*> إلى أقرب رقم عشري بعد العلامة العشرية ويضع بعد العلامة العشرية عددا من الأرقام مساويا للرقم الموجود في <*expN2*> . إذا كان الاختيار <*expN2*> يشتمـل على رقم سالب فسيتم تقريب الرقم الصحيح نفسه.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

```
? ROUND(43.457,2)          && 43.46   الاجابة
? ROUND(43.457,0)          && 43      الاجابة
? ROUND(43.457,1)          && 43.5    الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

INT( )

الوظيفة ROW()

تعطي رقم السطر الذي يقف عنده مؤشر الشاشة.

الشكل العام:

ROW()

الشـــرح:

تستخدم هذه الوظيفة غالبا داخل البرنامج لتحديد مكان السطر الذي يجب أن يظهر عنده مؤشر الشاشة وبإضافة أو طرح رقم معين من ROW() يمكن توجيه المؤشر إلى سطر آخر.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

لكي تظهر عبارة "Do you want to skip more" بعد خمسة أسطر من مكان المؤشر أدخل الأمر التالي:

```
@ ROW( ) + 5,1 SAY "Do you want to skip more"
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@ – COL( ) – PCOL( ) – PROW( )

الوظيفة SAVESCREEN( )

تحفظ جزء من الشاشة داخل الذاكرة تمهيدا لاسترجاعه.

الشكل العام:

SAVESCREEN (<*expN1*>, <*expN2*>, <*expN3*>, <*expN4*>)

حيـــث:

<*expN1*>...<*expN4*> : أرقام تحدد الجزء المطلوب حفظه بالترتيب التالي: أقصى اليسار السلفى، أقصى اليمين السلفى.

الشـــرح:

تسمح هذه الوظيفة بحفظ جزء من الشاشة الحالية تمهيدا لاسترجاع هذا الجزء بالوظيفة RESTSCREEN( ) . في حين يسمح أمر SAVE SCREEN بحفظ محتويات الشاشة كلها وأمر RESTORE SCREEN باسترجاع محتويات الشاشة كلها.

الاختلاف عن *dBASE III PLUS*: لا توجد بها.

مثـــال:

راجع المثال الموجود بالوظيفة RESTSCREEN( ) .

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

RESTSCREEN( )

الوظيفة SECONDS( )

تعطي رقما يقابل الوقت الموجود بالحاسب بالثانية.

الشكل العام:

SECONDS( )

الشـــرح:

تعطي هذه الوظيفة رقما يقابل الوقت الموجود بالحاسب بالثانية وتحسب بداية اليوم من الساعة الثانية عشرة من منتصف الليل. ولـذلك فالرقم الذي يمكن الحصول عليه يقع بين صفر و ٨٦٣٩٩ وهي الثواني الموجودة في ٢٣ ساعة + ٥٩ دقيقـة + ٥٩ ثانية. وتستخدم هذه الوظيفة إذا أردت أن تحسب عدد الثواني التي يستغرقها أمر معين مثل INDEX أو UPDATE .

الاختلاف عن *dBASE III PLUS* : غير موجودة بها.

مثـــال:

المثال التالي يعدل رصيد المخازن بعد عملية البيع بطرح الكمية المباعة من الكمية الموجودة بالمخازن ولأن أمر UPDATE يستغرق وقتا طويلا نسبيا فإننا نحسب عدد الثواني التي استغرقها تنفيذ الأمر ثم نظهر رسالة بعدد هذه الثواني.

```
SELECT 2
USE SALE  INDEX SALENO
SELECT 1
USE INVENT INDEX INVNO
BEGSEC = SECOND()
UPDATE ON ITEM_NO FROM B REPLACE ON_HAND WITH ON_HAND - B->QTY_SOLD
? "Processing update took " + (STR(SECONDS() _ BEGSEC) + "Seconds"
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

TIME( )

الوظيفة SELECT( )

تحدد اسم المنطقة المختارة.

الشكل العام:

SELECT ([<*expC*>])

حيـــث:

<*expC*> : الاسم البديل (Alias) للمنطقة التي تسأل عنها.

الشـــرح:

تعطي هذه الوظيفة رقما يقابل رقم المنطقة التي تسأل عنها ويقع هذا الرقم بين صفر و٢٥٥، وهذه المنطقة التي تسأل عنها إما أن تكون المنطقة الحالية (Current work area) وفي هذه الحالة لن تحتاج للاختيار <*expC*> أو تكون منطقة غير الحالية وفي هذه الحـالة يجب أن تستخدم الاختيار <*expC*> وهو عبارة عن حرف أو اسم يدل على المنطقة المطلوبة.

هـام: يجب ألا تخلط بين أمر SELECT(<*expN*>) والوظيفة SELECT(<*expC*>). فالأمر SELECT يستخدم الأقواس كوضع استثنائي للاشارة إلى رقم المنطقة. أما الأقواس في الوظيفة SELECT( ) فهي تستخدم لتشتمل على معطيات الوظيفة شأنها شأن أي وظيفة أخرى.

الاختلاف عن *dBASE III PLUS* : غير موجودة بها.

مثـــال:

في المثـال التالي تظهر رسالة تعطي رقم المنطقة المختارة بعد فتح كل مُلف ليسهل التعرف عليها أثناء التنفيذ في مرحلة اختبار البرنامج.

```
SELECT 2
USE SALE  INDEX SALENO
? "Current work area:", SELECT()     && 2   الإجابة
SELECT 1
USE INVENT INDEX INVNO
? "Current work area:", SELECT()     && 1  الإجابة
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

SELECT – ALIAS( )

الوظيفة SETCANCEL( )

تسمح أو تمنع استخدام مفتاح Alt-C للخروج من النظام.

الشكل العام:

SETCANCEL ([<expL>])

حيـــث:

<expL> : عبارة منطقية: .T. أو .F.

الشـــرح:

تشبه هذه الوظيفة أمر SET ESCAPE الموجود في «دي بيس ثري بلاس». وتـزيد عنـه أنها تعـطي قيمـة منـطقية (.T. أو .F.) وهـذه القيمـة يمكن تخزينها واسترجاعها. إذا اشتملت الوظيفة على القيمة .T. فسيتاح استخدام مفتاح Alt-C للخروج من البرنامج. والعكس صحيح. وننصح باستخدام هذه الوظيفة بعد انتهاء تجربة النظام وتسليمه للعميل لتمنع المستخدم من الخروج من البرنامج قبل إغلاق الملفات أو أثناء تعديل الملفات.

الاختلاف عن *dBASE III PLUS*: غير موجودة بها.

مثـــال:

لتمنـع المستفيد من الخروج من النظام أثناء تنفيذه إذا ضغط مفتاح Alt-C استخدم هذا الأمر

SETCANCEL(.F.)

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

SET ESCAPE – ALTD( )

الوظيفة SETCOLOR( )

تعطي معلومات عن الألوان المختارة أو تتيح تغيير الألوان.

الشكل العام:

SETCOLOR ([<*expC*>])

حيـــث:

<*expC*> : تعبير يحدد الألوان المختارة كما يفعل أمر SET COLOR TO .

الشـــرح:

تسمـح هذه الـوظيفـة بحفظ واسترجاع الألوان وتعطي رسالة تحدد الألوان المختـــارة بأمـــر SET COLOR (راجـع أمـر SET COLOR). ولـذلـك فيمكن استخدامها في برنامج فرعي لتغيير الألوان المختارة وإرجاع الألوان السابقة المختارة لباقي النظام.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثـال التـالي يستخدم الوظيفة SET COLOR( ) لتخزين اختيارات الألوان الحالية ولاختيار ألوان جديدة للشاشة.

```
STORE SETCOLOR() TO OLD_CLR   && ضع الالوان الحالية بالذاكرة فى حقل
                              && OLD_CLR اسمه
STORE "W+/B+,GR+/R+" TO NEW_CLR
ACT_CLR = SETCOLOR(NEW_CLR)
? ACT_CLR                     && W+/B+,GR+/R+  الاجابة
? OLD_CLR                     && ستحصل على تعريفات الالوان المحفوظة
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

SET COLOR

الوظيفة SETPRC()

تحدد قيمة لكل من الوظيفة PROW() والوظيفة PCOL() .

الشكل العام:

SETPRC (<*expN1*>, <*expN2*>)

حيث:

<*expN1*> : رقم أو تعبير رقمي يدل على السطر.

<*expN2*> : رقم أو تعبير رقمي يدل على العمود.

الشرح:

لأن هذه الوظيفة تحدد رقم السطر والعمود الذي ستبدأ عندهما الطابعة الطباعة المطلوبة فهي تستخدم لتوجيه الطابعة لتبدأ من أول الصفحة بدون أمر EJECT إذا استخدمناها بهذا الشكل: SETPRC(0,0)

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثال:

المثال التالي يطبع بيانات كل طالب من بداية صفحة جديدة برغم أن بيانات الطالب تحتل عشرين سطرا من كل صفحة.

```
USE STUDENTS
SET DEVICE TO PRINT
DO WHILE .NOT. EOF()
    @ 1,1 SAY TRIM(FIRSTNAME)+" "+LASTANME
    @ 2,1 SAY ADDRESS
    @ 3,1 SAY CITY
    * More @...SAY
    SETPRC(0,0)
    SKIP
ENDDO
SET DEVICE TO SCREEN
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

PROW( ) – PCOL( )

الوظيفة SOUNDEX() ٥

تحول عبارة حرفية إلى أربعة حروف تستخدم لتعامل أي عبارة قريبة منها في النطق معاملة هذه العبارة عند البحث عنها أو فهرستها.

الشكل العام:

SOUNDEX(<*expC*>)

حيـــث:

<*expC*> : العبارة المطلوب تحويلها.

الشـــرح:

تستخـدم هذه الوظيفة في الحالات التي يلزم فيها البحث عن عبارة ولكننا لسنا متأكدين من النطق الصحيح لها ونحتاج للبحث عنها بعبارة أخرى قريبة منها في النطق.

والذي يحدث أن هذه الوظيفة تأخذ العبارة أو الكلمة الأصلية وتحولها إلى عبارة أو كلمة جديدة تشتمل على حرف واحد وثلاثة أرقام ويمكن استخدام هذه الوظيفة مع أمـر SEEK أو FIND للبحث عن عبارة قريبة في النطق من تلك الموجودة في الملف. كما يمكن استخدامها عند الفهرسة لنجعل أمر INDEX يأخذ في اعتباره الكلمات القريبة في النطق ليرتبها بجوار بعضها.

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال ١ :

شركة طيران تتلقى حجوزات الركاب هاتفيا. ولأن موظف الحجز قد يخطىء في كتابة الاسم. ولمنع حدوث الخطأ عند الرد على الراكب من قبل موظف آخر يمكن استخدام المثال التالي عند فهرسة ملف الركاب.

```
USE PASSENGER
INDEX ON SOUNDEX(LASTNAME) TO I_PASS
```

مثـــال ٢ :

المثال التالي يبحث عن اسم الراكب بتحديد الاسم القريب منه في النطق.

```
M_PASS = SPACE(12)
@ 12,10 SAY "Enter passenger name: " GET M_PASS
READ
SEEK SOUNDEX(M_PASS)
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

FIND – SEEK – LOCATE – INDEX – SET SOFTSEEK

SPACE() الوظيفة

تنشيء عبارة حرفية تتكون من عدد من الفراغات.

الشكل العام:

SPACE(<*expN*>)

حيــث:

<*expN*> : عدد الفراغات المطلوبة.

الشــرح:

تستخدم هذه الوظيفة لانشاء حقل أو عبارة حرفية تشتمل على عدد من الفراغات. ويجب ألا يزيد عدد الفراغات المطلوبة (<*expN*>) على ٦٤ ك.ب.

ويمكن الاستفادة من هذه الوظيفة في حالة البحث عن معلومة داخل الملف مثل رقم الموظف أو رقم المخزون بإظهار عدد من الفراغات أمام المستخدم ليدخل الرقم المطلوب ويتم البحث في الملف بناء على الرقم الذي أدخل ويتضح ذلك من المثال الآتي.

الاختلاف عن *dBASE III PLUS* : تسمح «دي بيس ثري بلاس» بحد أقصى قدره ٢٥٤ فراغا فقط.

مثــال:

المثال الآتي يبحث في ملف STOCK.DBF عن اسم الشركة التي يدخلها المستخدم في حقل COMPANY المستخدم كمفتاح

```
USE STOCK INDEX ICOMP
STORE SPACE(30) TO M_COMP
@ 10,10 SAY "Enter company name" GET M_COMP
READ
SEEK M_COMP
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

REPLICATE( )

الوظيفة SQRT( )

تعطى الجذر التربيعي لرقم.

الشكل العام:

SQRT(<*expN*>)

حيـــث :

<expN> : رقم أو تعبير أو حقل رقمي.

الشـــرح :

تعـطى هذه الـوظيفـة الجـذر التربيعي للأرقام الموجبة فقط. ويظهر الجذر التربيعي مشتملا على رقمين عشريين (القيمة التلقائية) أو عدد من الأرقام الغشرية مساويا للأرقام العشرية الموجودة في الرقم.

الاختلاف عن ***dBASE III PLUS*** : لا يوجد اختلاف.

مثـــال :

```
? SQRT(100)          && 10.00    الاجابة
? SQRT(100.000)      && 10.000   الاجابة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET DECIMAL – SET FIXED

الوظيفة STR()

تحول عبارة رقمية إلى عبارة حرفية.

الشكل العام:

STR(<*expN*> [,<*length*>][,<*decimal*>])

حيـــث:

<*expN*> : الرقم أو العبارة الرقمية المطلوب تحويلها.

<*lenth*> : الطول المطلوب للرقم.

<*decimal*> : عدد الأرقام بعد العلامة العشرية.

الشــــرح:

تحول رقم أو عبـارة رقمية إلى عبـارة حرفية. وتـلاحظ أن وضـع كلا من الاختيارين <*length*> و<*decimal*> اختياري في الوظيفة. فإذا لم تشتمل الوظيفة على <*length*> فستخصص قاعدة البيانات طول قدره ١٠ حروف. وإذا لم تشتمل على <*decimal*> فإن الرقم يتم تقريبه لأقرب رقم صحيح.

ويجب الانتباه إلى أن الطول المحدد في الأمر (<*length*>) يشتمل على إجمالي عدد الأرقـام النـاتجـة من الـوظيفة (الرقم الصحيح + العلامة العشرية + الأرقام العشرية) وإلى أن الرقم الذي يحدد عدد الأرقام العشرية (<*decimal*>) يشتمل على عدد الأرقـام التي ستـظهـر بعـد العلامة العشرية. فإذا حددت عددا من الأرقام العشرية يقل عن الموجود في العبارة الرقمية فسيظهر لك عدد الأرقام العشرية المحددة فقط مقربة لأقرب رقم عشري.

إذا اشتملت الوظيفة على أقل من عدد الأرقام الصحيحة الموجودة في <*expN*> فستعطيك قاعدة البيانات علامة ** بدلا من الرقم.

الاختلاف عن *dBASE III PLUS* : استخدام هذه الوظيفة مع DAY( ) أو

( )MONTH أو ( )YEAR في «كلبر» لا يتبع القواعد العامة السابقة. فمثلاً:
(( )STR(YEAR تعطي ٥ خانات بينما تعطي كل من (( )STR(DAY و
(( )STR(MONTH ٣ خانات فقط.

مثـــال:

المثال الآتي يستخدم الوظيفة ( )STR مع كل الحالات السالفة:

```
? STR(5656.676,8,3)        && 5656.676    الاجابة
? STR(5656.676,8,2)        && 5656.68     الاجابة
? STR(5656.676)            && 5657        الاجابة
? STR(5656.676,2)          && **          الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

VAL( ) – TRANSFORM( )

الوظيفة STRTRAN( )

تقوم بالبحث عن عبارة حرفية داخل أخرى واستبدالها بقيمة جديدة.

الشكل العام:

STRTRAN(<expC1>, <expC2> [,<expC3>] [,<expN1>][,<expN2>])

حيـــث:

<expC1>	:	العبارة التي سيتم البحث فيها.
<expC2>	:	العبارة المطلوب البحث عنها.
<expC3>	:	العبـــارة التي ستحـــل محل القـــديمـــة. إذا اهمـــل هذا الاختيار فستحـــذف كل الحـــروف أو الكلمات المتـــطابقة مع العبارة التي نبحث عنها.
<expN1>	:	أول عبـــارة أو كلمة سيتم استبدالها إذا وجدت. إذ أهمل هذا الاختيار فستستبدل أول عبارة تتطابق مع العبارة التي نبحث عنها.
<expN2>	:	عدد الكلمات أوالعبارة التي سيتم استبدالها في حالة مطابقتها مع العبارة التي نبحث عنها. وإذا أهمل هذا الاختيار فسيتم استبدال كل الكلمات أو العبارات التي تتطابق مع عبارة البحث (<expC2>).

الشـــرح:

من الشرح السابق لاختيارات هذه الوظيفة نعرف أنها تقوم بالبحث عن كلمة أو عبارة داخل عبارة أو حقل حرفي ثم تستبدل كل أو بعض العبارات التي تتطابق مع عبارة البحث بعبارة جديدة.

الاختلاف عن dBASE III PLUS : لا توجد بها.

مثـــال:

في المثال التالي تم إدخال علامة النقطة (.) قبل وبعد كلمة .St في حقل ADDRESS أثنـاء إدخـال بيانات الطلاب في ملف STUDENTS والمطلوب فقط النقـطة التي تلي كلمة St. المثال التالي يستخدمْ الوظيفة ()STRTRAN لتصحيح هذا الخطأ. لاحظ محتويات الحقل قبل التصحيح وبعد التصحيح.

قبل التصحيح : 20 Airport .St. Cairo

الأمر اللازم للتصحيح

```
REPLACE ALL ADDRESS WITH  STRTRAN(ADDRESS,".","  ",1,1)
```

ويمكن ترجمة هذا الأمر كما يلي: ابحث في حقل ADDRESS (*<expC1>*) عن علامة النقطة (*<expC>*) تم استبدلها بفراغ.

ابتداء من أول علامة تجدها (*<expN1>*) واستبدل علامة واحدة فقط (*<expN2>*) . وستظهر العبارة بعد التصحيح هكذا:

20 Airport .ST. Cairo

المكتبة: **CLIPPER.LIB**

الأوامر ذات الصلة:

AT – RAT( ) – SUBSTR( )

الوظيفة STUFF( )

تستبدل حرف أو حروف موجودة في عبارة حرفية بحرف أو حروف أخرى.

الشكل العام:

STUFF(<*expC1*>, <*expN1*>, <*expN2*>, <*expC2*>)

حيـــث:

<*expC1*> : العبارة الحرفية الأساسية.

<*expN1*> : بداية الحروف التي سيتم استبدالها داخل العبارة الأساسية.

<*expN2*> : عدد الحروف التي ستتبدل من العبارة الأساسية.

<*expC2*> : العبارة الجديدة التي ستحل محل العبارة الأساسية.

الشـــرح:

تسمح لك هذه الوظيفة باستبدال حروف العبارة الأساسية (<*expC1*>) حرفا حرفا ابتداء من المكان المحدد في <*expN1*> بالعبارة أو الحروف الموجودة في العبارة الثانية (<*expC2*>) .

إذا كانت قيمة (<*expC2*>) صفر فإن العبارة الثانية (<*expC2*>) ستدخل في العبارة الأولى بدون حذف أية حروف.

إذا لم تشتمل العبارة الثانية (<*expC2*>) على شيء فإن عدد الحروف المحدد في الأمر سيحذف من العبارة الأساسية بدون إحلال حروف أخرى مكانها.

ملاحظة: إذا كانت العبارة الأساسية حقل حرفي موجود في ملف قاعدة بيانات فلن تتأثر محتويات الحقل بهذه الوظيفة فإذا أردت استبدال بيانات الحقل نفسه فيجب أن تستخدم أمر REPLACE .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

أمثلـــة:

المثال الآتي يستبدل كلمة بأخرى

```
OLD = "I lost data"
NEW = "find"
? STUFF(OLD,3,4,NEW)                    && I find data   الاجابة
```

والمثـال التالي يدخل العبارة الثانية (<*expC2*>) في العبارة الأولى ابتداء من الحرف الثامن وهو حرف d بدون حذف شيء منها:

```
? STUFF("I find data",8,0,"my ")     && I find my data   الاجابة
```

والمثـال التالي يحذف عدد الحروف المحدد في الأمر ابتداء من الحرف الثامن بدون إحلال حروف أخرى لأن <*expC2*> لاتشتمل على شيء:

```
? STUFF("I find my data",8,3,"")    && I find data    الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

LEFT( ) – RIGHT( ) – SUBSTR( )

الوظيفة SUBSTR( )

تستخرج جزء من عبارة حرفية.

الشكل العام:

SUBSTR(<*expC*>, <*starting position*> [,<*no. of characters*>])

حيــث:

<*expC*>	:	عبارة أو حقل حرفي.
starting position	:	بداية العبارة الفرعية (substring)
no. of characters	:	عدد حروف العبارة الفرعية.

الشــرح:

تستخدم هذه الوظيفة لاستخراج عدد من الحروف (no. of characters) موجودة في عبارة حرفية (<*expC*>) ابتداء من حرف معين (starting position).

فإذا لم تحدد في الأمر عدد الحروف المطلوب استخراجها من العبارة (<*no. of characters*>) فإن قاعدة البيانات ستستخرج باقي العبارة ابتداء من الحرف المحدد (<*starting position*>).

الاختلاف عن dBASE III PLUS : في «كلبر» يمكن أن يكون <*starting position*> رقما سالبا ليوضح لكلبر أن العبارة الحرفية سيبدأ استخراجها من نهاية العبارة.

مثـــال:

```
STORE "5/31/1951" TO BIRTHDATE
? SUBSTR(BIRTHDATE,3,2)          &&   31  الاجابة
? SUBSTR(BIRTHDATE,-1,4)         && 1951  الاجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

AT( ) – LEFT( ) – RIGHT( ) – STR( ) – STUFF( )

الوظيفة TIME()

تظهر الوقت الحالي كما هو مسجل بالحاسب.

الشكل العام:

```
TIME( )
```

الشـــرح:

يظهر الوقت المسجل بالحاسب بالشكل الآتي:

```
hh–mm–ss
```

وتعتبر قاعدة البيانات أن الأرقام الدالة على الوقت عبارة حرفية ولذلك يجب أن تحول الساعات أو الدقائق أو الثواني المطلوبة لاجراء عمليات حسابية إلى أرقام أولا قبل إجراء العمليات الحسابية مثل طرح وقت من وقت.

الاختلاف عن dBASE III PLUS : لا يوجد اختلاف.

مثـــال:

لمعرفة الوقت المسجل بالحاسب

```
? TIME( )
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة:

```
SECOND( )
```

الوظيفة TONE( )

تسمح بسماع صوت الجرس بنغمات مختلفة.

الشكل العام:

TONE(<expN1> , <expN2>)

حيـــث:

<expN1> : معدل تحويل النغمة إلى صوت (يتحكم في نغمة الجرس).

<expN1> : مدة سماع النغمة. تحسب بأجزاء كل منها ١٨/١ من الثانية.

الشــــرح:

تستخدم هذه الوظيفة لاطلاق نغمة في حالات معينة لتنبيه المستخدم لخطأ ما أو غيره ولا يمكن الحصول على أصوات موسيقية لكن يمكن التحكم في نغمة الجرس ومدتها باستخدام اختيارات هذه الوظيفة. لاحظ أن <expN1> رقم يقع بين ٢٠ - ١٧٠٠٠ أما <expN2> فهي رقم يقع بين ١-١٨.

الاختلاف عن dBASE III PLUS : لا توجد بها.

مثـــال:

المثال التالي يستخدم هذه الوظيفة لسماع صوت الجرس بأنغام مختلفة.

```
TONE(100,5)
TONE(40,10)
TONE(5000,3)
TONE(2000,5)
```

المكتبة: EXTEND.LIB

الأوامر ذات الصلة:

SET BELL - CHR( )

الوظيفة TRANSFORM()

تسمح بتغيير شكل بيانات حقل أو عبارة.

الشكل العام:

TRANSFORM(<*expl*>, <*expC*>)

حيـــث:

<*expl*> : الحقل أو العبارة المطلوب تغييرها.

<*expC*> : الشكل المقترح.

الشـــرح:

تسمح هذه الوظيفة بتغيير شكل عبارة حرفية أو رقمية بشكل (Format) مغاير لما هي عليه. وذلك بدون حاجة لاستخدام أمر SAY...@وهي تستخدم مع أوامر الاظهار مثل ?/?? أو DISPLAY أو LIST أو LABEL أو REPORT .

ويشتمل الاختيار <*expl*> على العبارة الحرفية أو الرقمية المطلوب تغيير شكلها عنـد إظهـارهـا. بينما يشتمل الاختيار <*expC*> على الشكل المقترح لاظهار العبارة ويستخدم فيه الصور المسموح بها في الاختيار PICTURE مع أمر SAY...@ .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــال:

لاظهار الرقم 12345.67 بالشكل: 12,345.67

```
? TRANSFORM(12345.67,"99,999.99")
```

ولاظهار عبارة ''last name'' بالحروف الكبيرة

```
? TRANSFORM("last name", "@!")
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

STR( ) – @...SAY...GET – LOWER – UPPER( )

الوظيفة TRIM/RTRIM( )

تحذف جميع المسافات الموجودة على يمين عبارة أو حقل حرفي.

الشكل العام:

RTRIM(<*expC*>) أو TRIM(<*expC*>)

حيـث:

<*expC*> : عبارة أو حقل حرفي.

الشـرح:

تستخدم هذه الوظيفة لحذف المسافات الخالية الموجودة على يمين عبارة أو حقل حرفي. فمثلا إذا حددت عند بناء الملف أن طول الاسم الأول هو ١٤ حرفا وتم إدخال اسم ALI فسيحتل الاسم الحروف الثلاثة الأولى من الحقل في حين تبقى ٩ فراغات على يمين الاسم داخل نفس الحقل. وعند إظهار هذا الحقل فستظهر الفراغات التسعة بعد كلمة ALI وتبدو هذه المشكلة بوضوح عند استخدام أمر مثل LIST أو DISPLAY ولحذف هذه الفراغات التسعة استخدم هذه الوظيفة ليظهر الحقل التالي ـ وليكن اسم العائلة ـ بعد كلمة ALI مباشرة.

لاحظ أنك إذا استخدمت هذه الوظيفة متبوعة بعلامة "," فستفصل لك قاعدة البيانات بفراغ واحد بين الحقل والحقل التالي. أما إذا استخدمتها متبوعة بعلامة + فلن تفصل قاعدة البيانات بين الحقل والحقل التالي بأية فراغات.

الاختلاف عن *dBASE III PLUS*: لا يوجد اختلاف.

مثـال:

المثال الآتي يستخدم الوظيفة RTRIM( ) لحذف المسافات الموجودة على يمين حقل FIRSTNAME في ملف STUDENTS.DBF وذلك باستخدام أمر ? .

[redacted]

```
USE STUDENTS
GO 15
? RTRIM(FIRSTNAME),LASTNAME        && NASER MURAD   الإجابة
? RTRIM(FIRSTNAME)+LASTNAME        && NASERMURAD    الإجابة
? RTRIM(FIRSTNAME)+" "+LASTNAME    && NASER MURAD   الإجابة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

LTRIM( )

الوظيفة TYPE()

تعطي إجابة تدل على نوع البيانات المخزنة في العبارة.

الشكل العام:

TPYE (<*expC*>)

حيـــث:

<*expC*> : عبارة حرفية أو حقل حرفي.

الشـــرح:

تقيم هذه الـوظيفة العبارة الموجودة بها (<*expC*>) وتعطي إجابة من حرف واحد يدل على نوع البيانات المخزنة في هذه العبارة. وهذه الاجابة إما أن تكون C بمـعـنـى حرفي Character أو N بمـعـنـى رقمي Numeric أو D بمعنى تاريخي Date أو L بمعنى منطقي Logic أو M بمعنى Memo أو A بمعنى Array . فإذا لم تجد قاعدة البيانات الحقل أو العبارة في الملف أو في الذاكرة فستعطيك الاجابة U بمعنى غير معروف Undefined .

ويجب أن توضع <*expC*> بين علامتي تنصيص "".

الاختلاف عن *dBASE III PLUS* : لا تعطي «دي بيس ثري بلاس» بعض أنواع البيانات التي تستخدمها «كلبر».

مثـــال:

```
STORE "Morning" TO TXT
? TYPE("TXT")                    && C    الاجابة
STORE .T. TO MALE
? TYPE("MALE")                   && L    الاجابة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

FIELD( ) – STORE

الوظيفة UPDATED( )

تحدد هل تم تعديل محتويات حقل أثناء استدعاء أمر READ أم لا؟

الشكل العام:

UPDATED( )

الشـــرح:

عندما يتم استدعاء أمر READ لتعديل محتويات حقل أو متغير فإن الوظيفة UPDATED( ) تشتمل على القيمة .F. فإذا عدلت محتويات الحقل فإنها تشتمل على القيمة .T. ولذلك فيمكنك استخدام هذه الوظيفة لمعرفة هل تعدلت محتويات الحقل أو المتغير أثناء استدعاء أمر READ أم لا؟

الاختلاف عن *dBASE III PLUS* : لا توجد بها.

مثـــال:

المثـال التـالي يستبدل محتويات الحقول في حالة تعديل حقول الذاكرة أثناء استدعائها بأمر READ

```
M_SALARY = 0
* <more memvars>
@ 2,2 SAY "Enter new salary" GET M_SALARY PICT "99,999,.99"
* <more GETs>
IF UPDATED()
  REPLACE SALARY WITH M_SALARY
ENDIF
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

@...GET – READ

الوظيفة UPPER( )

تحول الحروف الصغيرة (Lower Case) إلى كبيرة (Upper Case) .

الشكل العام:

UPPER(<*expC*>)

حيث:

<*expC*> : حقل أو عبارة حرفية .

الشرح:

تقوم هذه الوظيفة بتحويل الحروف الانجليزية الصغيرة (Lower Case Letters) إلى حروف كبيرة (Upper Case Letters) .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثال:

المثال التالي يستخدم هذه الوظيفة:

```
STORE "good morning" TO DAY
?UPPER(DAY)                 && GOOD MORNING  الإجابة
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

LOWER( ) – ISUPPER( ) – ISLOWER( )

الوظيفة USED( )

تحدد هل ملف قاعدة البيانات مفتوح في المنطقة الحالية أم لا؟

الشكل العام:

USEED( )

الشـــرح:

تعطي هذه الوظيفة القيمة المنطقية .T. إذا كان الملف مفتوحا في المنطقة الحالية. وإلا فإنها تعطي وتفيد هذه الوظيفة في حالة شبكة الاتصالات المحلية للتأكد أن الملف فتح إذا كان مستفيد آخر في الشبكة قد أغلقه.

الاختلاف عن dBASE III PLUS : لا توجد بها.

مثـــال:

المثال التالي يحاول فتح الملف مائتين مرة فإذا فُتح قبل ذلك فإنه يخرج من دوارة المحاولة.

```
SET EXCLUSIVE OFF
FOR CNT = 1 TO 200
  USE STOCK
  IF USED()
    EXIT
  ENDIF
NEXT
```

المكتبة: CLIPPER.LIB

الأوامر ذات الصلة:

USE( ) – NETERR( )

الوظيفة VAL()

تحول عبارة حرفية تشتمل على أرقام إلى عبارة رقمية.

الشكل العام:

VAL(<*expC*>)

حيــــث:

<*expC*> : عبارة حرفية تشتمل على أرقام.

الشـــــرح:

تستخدم هذه الوظيفة لتحويل عبارة حرفية تشتمل على أرقام أو أرقام سبق إدخالها إلى حقل حرفي إلى عبارة رقمية أو إلى أرقام يمكن إجراء عمليات حسابية عليها.

ولذلك، فإذا اشتملت العبارة المطلوب تحويلها على حروف قبل الأرقام أو على حروف فقط فإن ناتج هذه الوظيفة سيكون صفرا.

إذا اشتملت على فراغات قبل أو بعد العبارة الرقمية أو الرقم فإن قاعدة البيانات تهمل هذه الفراغات وتعتبر أن أول فراغ على يمين الرقم يعني نهاية الرقم.

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف.

مثـــــال:

لاحظ تأثير VAL( ) في الأمثلة الآتية:

```
NUM = "12345.67"
? TYPE("NUM")              && C  الاجابة
STORE VAL(NUM) TO NUM1

? TYPE("NUM1")             && N  الاجابة
? VAL("ABC")               && 0  الاجابة
```

المكتبة : **CLIPPER.LIB**

الأوامر ذات الصلة :

SET DECIMALS – SET FIXED – STR( )

الوظيفة YEAR( )

تظهر رقم يدل على السنة .

الشكل العام:

YEAR(<*expD*>)

حيــــث :

<*expD*> : حقل أو عبارة تاريخية .

الشــــرح:

تظهر رقم مكون من ٤ خانات للدلالة على السنة المسجلة بحقل تاريخي أو عبارة تاريخية .

الاختلاف عن *dBASE III PLUS* : لا يوجد اختلاف .

مثـــال :

للحصـول على تاريخ اليوم بحيث يظهـر اسم اليوم متبوعا بترتيب اليوم في الشهر ثم الشهر ثم السنة :

```
? CDOW(DATE()),DAY(DATE()),CMONTH(DATE()),YEAR(DATE())
* Monday 29 October 1990 ستحصل على الاجابة التالية:
```

المكتبة : CLIPPER.LIB

الأوامر ذات الصلة :

DATE( ) – DAY( ) – MONTH( )

الملاحق

الملحق الأول   ملخص المصطلحات والرموز

الملحق الثانى   الشفرة الامريكية القياسية لتبادل المعلومات (ASCII)

الملحق الثالث   أهم المكتبات والبرامج التى يمكن الاستفادة منها في إعداد نظم ادارة قواعد البيانات

الملحق الأول

ملخص المصطلحات والرموز

يشتمل هذا الملحق على المصطلحات والرموز المستخدمة في أوامر ووظائف «كلبر»

الرمز أو المصطلح	المعنى
[ ]	هذه الأقواس تعني أن ما بداخلها اختياريا ولكن الأقواس نفسها لا تكتب ضمن الأمر.
/	هذه العلامة تعني أحد اختيارين إما أن تختار الموجود قبلها أو الموجود بعدها أما العلامة نفسها فلا تكتبها فمثلا (FOR/WHILE) إما أن تستخدم FOR أو WHILE .
< >	هذه الأقـواس تعني أن ما بداخلها إلزامي أي يجب أن يكتب ضمن الأمر والأقواس نفسها لا تكتب ضمن الأمر.
Condition	تحدد لك الحالات التي ينفذ فيها هذا الأمر تحديدا تاما فمثلا "CITY = "RIYADH معناها تنفيذ الأمر كلما وجد كلمة RIYADH في حقل CITY .
Scope	يحدد إلى أي مدى يجب أن ينفذ الأمر فمثلا : لتنفيذ الأمر مع السجلات العشرة التالية : NEX 10 لتنفيذ الأمر مع باقي السجلات : REST لتنفيذ الأمر مع كل السجلات : ALL لتنفيذ الأمر مع السجل رقم ٣ : RECORD 3 وإذا لم تحدد في الأمر فتأخذ قاعدة البيانات القيمة التلقائية (Default value) . وهي كل السجلات .
Prompt	عبارة أو جملة حرفية تختارها لتظهر أثناء تنفيذ بعض الأوامر كما هي فمثلا "TEST" تظهر على الشاشة TEST .

الرمز أو المصطلح	المعنـــــــــى
memvar	حقل ذاكرة (memory variable) .
memvar list	حقل أو مجموعة حقول موجودة بالذاكرة .
expression	حقل موجود في الملف (data field) أو حقل موجود بالذاكرة (memory variable) أو تعبيرا حرفيا أو رقميا أو تاريخيا أو منطقيا تقبله قاعدة البيانات .
exp	اختصار لكلمة expression .
expC	تستخدم للدلالة على تعبير حرفي Character expression .
expN	تستخدم للدلالة على تعبير رقمي Numeric expression .
expD	تستخدم للدلالة على تعبير تاريخي Date expression .
expL	تستخدم للدلالة على تعبير منطقي Logical expression
expression list	حقل أو مجموعة حقول موجودة إما بالملف أو بالذاكرة أو عبارات خارجية مفصولة بعلامة « , » وقد تشتمل على احدى العلامات الحسابية .
template	مجموعة حروف تنظم شكل الطباعة .
field	حقـل . مساحة محددة كجزء من سجل (record) تستخدم لتخزين جزء من المعلومات فمثلا رقم الطالب أو اسم الطالب يعتبر جزءا من سجل الطالب في ملف الطلاب . ويعبر عنه بالحقل .
fieldname	الاسم الذي يعطى للحقل داخل السجل .
field list	إما اسم حقل أو أسماء مجموعة حقول موجودة في الملف .
field type	معناه نوع البيانات المخزنة داخل السجل وهي إما حرفية (C) أو رقمية (N) أو تاريخية (D) أو منطقية (L) أو ملاحظات (M) .
file	مجموعة سجلات متصلة ببعضها تحت اسم ملف مسجل على قرص ممغنط .

الرمز أو المصطلح	المعنـــــى
file name	اسم ملف موجود على القرص.
record	سجل. مجموعة حقول تكوّن وحدة معلومات داخل الملف مثل المعلومات المتوفرة عن طالب معين وتشتمل على اسمه وعنوانه ورقمه.. الخ.
Commands	أمر أو مجموعة أوامر تستخدمها قاعدة البيانات Clipper.
Key word	أمر أو كلمة معروفة لقاعدة البيانات Clipper مثل: ON أو GOTO.
key field	هو الحقل الذي يتم فهرسة أو ترتيب أو تعديل أو تجميع ملف قاعدة البيانات طبقا لبياناته في أوامر SORT – INDEX – JOIN – UPDATE – TOTAL
Character string	سلسلة من الرموز أو عبارة تشتمل على حروف أبجدية أو أرقام أو علامات. ولا يمكن إجراء عمليات حسابية على الأرقام التي تحتويها وعادة توضع بين علامتي تنصيص " " أو ' '.
Column	عمود. رقم يوضح مكان العمود الذي يقف عنده المؤشر على الشاشة أو الطابعة. والعمود هو خط رأسي لمجموعة من المواضع التي تخزن عليها المعلومات وعادة تقسم الشاشة إلى ٨٠ عمودا رأسيا تبدأ من صفر إلى ٧٩.
row	الصف أو السطر. رقم يوضح الخط الأفقي الذي يقف عنده المؤشر إما على الشاشة أو الطابعة وعادة تقسم الشاشة إلى ٢٥ سطرا تبدأ من صفر إلى ٢٤.
FOR Condition	كلمة FOR معناها التي تخص وتستخدم في معظم الأوامر اختياريا لتوضح لقاعدة البيانات أن الأمر يجب أن ينفذ في الحالات المحددة بعدها فقط.
WHILE Condition	كلمة WHILE معناها طالما وتستخدم في معظم الأوامر

الرمز أو المصطلح	المعنـــــى
file name	اسم ملف موجود على القرص.
record	سجل. مجموعة حقول تكوّن وحدة معلومات داخل الملف مثل المعلومات المتوفرة عن طالب معين وتشتمل على اسمه وعنوانه ورقمه. . الخ.
Commands	أمر أو مجموعة أوامر تستخدمها قاعدة البيانات Clipper .
Key word	أمر أو كلمة معروفة لقاعدة البيانات Clipper مثل: ON أو GOTO .
key field	هو الحقل الذي يتم فهرسة أو ترتيب أو تعديل أو تجميع ملف قاعدة البيانات طبقا لبياناته في أوامر SORT – INDEX – JOIN – UPDATE – TOTAL
Character string	سلسلة من الرموز أو عبارة تشتمل على حروف أبجدية أو أرقام أو علامات. ولا يمكن إجراء عمليات حسابية على الأرقام التي تحتويها وعادة توضع بين علامتي تنصيص " " أو ' '.
Column	عمود. رقم يوضح مكان العمود الذي يقف عنده المؤشر على الشاشة أو الطابعة. والعمود هو خط رأسي لمجموعة من المواضع التي تخزن عليها المعلومات وعادة تقسم الشاشة إلى ٨٠ عمودا رأسيا تبدأ من صفر إلى ٧٩.
row	الصف أو السطر. رقم يوضح الخط الأفقي الذي يقف عنده المؤشر إما على الشاشة أو الطابعة وعادة تقسم الشاشة إلى ٢٥ سطرا تبدأ من صفر إلى ٢٤.
FOR Condition	كلمة FOR معناها التي تخص وتستخدم في معظم الأوامر اختياريا لتوضح لقاعدة البيانات أن الأمر يجب أن ينفذ في الحالات المحددة بعدها فقط.
WHILE Condition	كلمة WHILE معناها طالما وتستخدم في معظم الأوامر

الملحق الثاني

الشفرة الأمريكية القياسية لتبادل المعلومات

American Standard Code for Information Interchange (ASCII)

يوضح الجدول التالي قيم الرموز والأرقام والأحرف والعلامات الخاصة المقابلة للشفرة الأمريكية (ASCII Code) مرتبة تصاعديا حسب القيمة العشرية المقابلة لكل حرف أو رقم أو رمز ويوضح الجدول كذلك معنى الأكواد من صفر إلى ٣١.

وتستخدم وظيفة ( )ASC في قاعدة البيانات للحصول على الشفرة المقابلة للحروف. فإذا أردت معرفة الشفرة (ASCII Code) المقابلة للحرف M أدخل الأمر التالي:

```
? ASC ("M")
```

ستحصل على الرقم 77

بينما تستخدم وظيفة ( )CHR لاظهار الحرف المقابل للشفرة (ASCII Code) التي تعطى له. فمثلا لمعرفة الحرف المقابل للكود 77 يجب أن تدخل الأمر الآتي:

```
? CHR (77)
```

ستحصل على حرف M

ASCII Value الشفرة	Character الحرف	Code الكود	Symbol الرمز
000	(null)	(null)	NUL
001	☺	Ctrl-A	SOH
002	●	Ctrl-B	STX
003	♥	Ctrl-C	ETX
004	♦	Ctrl-D	EOT
005	♣	Ctrl-E	ENO
006	♠	Ctrl-F	ACK
007	(beep)	Ctrl-G	(bell) BEL
008	■	Ctrl-H	(backspace) BS
009	(tab)	Ctrl-I	(tab) horizontal HT
010	(line feed)	Ctrl-J	(linefeed) LF
011	(home)	Ctrl-K	(vertical tabs) VT
012	(form feed)	Ctrl-L	(formfeed) FF
013	(carriage return	Ctrl-M	(carriage return) CR
014	♫	Ctrl-N	SO
015	☼	Ctrl-O	SI
016	►	Ctrl-P	DLE
017	◄	Ctrl-Q	DC1
018	↕	Ctrl-R	DC2
019	‼	Ctrl-S	DC3
020	¶	Ctrl-T	DC4
021	§	Ctrl-U	NAK
022	▬	Ctrl-V	SYN
023	↨	Ctrl-W	ETB
024	↑	Ctrl-X	CAN
025	↓	Ctrl-Y	EM
026	→	Ctrl-Z	SUB
027	←	Escape	
028	(cursor right)	FS	
029	(cursor left)	GS	
030	(cursor up)	RS	
031	(cursor down)	US	

ASCII Value الشفرة	Character الحرف
032	(space)
033	!
034	"
035	#
036	$
037	%
038	&
039	'
040	(
041	)
042	*
043	+
044	,
045	-
046	.
047	/
048	0
049	1
050	2
051	3
052	4
053	5
054	6
055	7
056	8
057	9
058	:
059	;
060	<
061	=
062	>
063	?
064	@
065	A
066	B
067	C
068	D
069	E
070	F
071	G
072	H
073	I
074	J
075	K
076	L
077	M
078	N
079	O
080	P
081	Q
082	R
083	S
084	T
085	U
086	V
087	W
088	X
089	Y
090	Z
091	[
092	\
093	]
094	^
095	_
096	`
097	a
098	b
099	c
100	d
101	e
102	f
103	g
104	h
105	i

ASCII Value الشفرة	Character الحرف
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	
125	}
126	~
127	⌂
128	Ç
129	u
130	é
131	â
132	ä
133	à
134	å
135	ç
136	ê
137	ë
138	è
139	ï
140	î
141	ì
142	A
143	Å
144	E
145	æ
146	Æ
147	ô
148	o
149	ò
150	û
151	ù
152	ÿ
153	O
154	U
155	¢
156	£
157	¥
158	Pt
159	ƒ
160	á
161	í
162	ó
163	ú
164	ñ
165	Ñ
166	ª
167	º
168	¿
169	⌐
170	¬
171	½
172	¼
173	¡
174	«
175	
176	
177	▒
178	▓
179	│

ASCII Value الشفرة	Character الحرف	ASCII Value الشفرة	Character الحرف
180	┤	218	┌
181	╡	219	█
182	╢	220	▄
183	╖	221	▌
184	╕	222	▐
185	╣	223	▀
186	║	224	α
187	╗	225	β
188	╝	226	Γ
189	╜	227	π
190	╛	228	Σ
191	┐	229	σ
192	└	230	μ
193	┴	231	τ
194	┬	232	Φ
195	├	233	Θ
196	─	234	Ω
197	┼	235	δ
198	╞	236	∞
199	╟	237	Ø
200	╚	238	(
201	╔	239	∩
202	╩	240	≡
203	╦	241	±
204	╠	242	≥
205	═	243	≤
206	╬	244	⌠
207	╧	245	⌡
208	╨	246	÷
209	╤	247	≈
210	╥	248	°
211	╙	249	•
212	╘	250	·
213	╒	251	√
214	╓	252	ⁿ
215	╫	253	²
216	╪	254	■
217	┘	255	(blank 'FF')

الملحق الثالث

نوضح فيما يلي اهم المكتبات والبرامج والتى يمكن الاستفادة منها عند اعداد نظم ادارة قواعد بيانات والتى تهم المبرمجين او المهتمين بتطوير برامجهم باستخدام «كلبر» واتماماً للفائدة فقد أوردنا أسماء وعناوين الشركات المنتجة لهذه البرامج والمكتبات

dCLIP

يوفر هذا البرنامج على مبرمجى «كلبر» وقت ترجمة وربط برامجهم في مرحلة تجارب النظام وقبل أن يصير النظام جاهزا للتشغيل. وذلك لأنه يحتوى على نقطة محث Dot Prompt مثل تلك الموجودة في «دى بيس» تمكن المبرمج من تجربة البرامج واختبارها قبل ربطها مع بعضها ومع نظام التشغيل. وكما هو معروف فان ٩٠٪ من وقت تطوير النظم يضيع في مراحل اختبار النظام التى تتشمل في تصحيح الاخطاء واعادة الترجمة والربط كلما اكتُشفَ خطأ بالبرنامج ويشتمل أيضا على مكتشف أخطاء (Debugger) يسهل كثيرا عملية تعقب واكتشاف الأخطاء. ومن مزاياه أيضا توافقه مع معظم برامج الربط مثل

plink 86 Plus. RTL, NK – BLINKER – WARPLINK

الشركة المنتجة:

DONNAY Software Designs
1880 Park Newport, Suit 104
Newport Beach, CA 92660

ARTFUL. LIB

مكتبة برامج جاهزة تشتمل على اكثر من ٣٥٠ وظيفة وتشبه مصمم التطبيقات (Applicotion generator) في تسهيل اعداد نظم ادارة قواعد البيانات. وهى توفر عليك حوالى ٩٠٪ من الأوامر التى يحتاجها نظام ادارة قواعد البيانات لأنها تحتوى على برامج جاهزة للأغراض الآتية:

ـ البحث عن معلومة في ملف واحد أو أكثر من ملف
ـ برامج تصحيح المدخلات (Validations)
ـ اعداد التقارير
ـ النوافذ
ـ شاشات المساعدة

الشركة المنتجة

Applications I nc.
176 St. George St., Dept. A2
Toronto, Ontario M 5R 2M7, Canada

Flipper 5.0

تتيح امكانية تمثيل بيانات «كلبر» برسوم بيانية بأشكال مختلفة وبامكانيات متعددة مثل اظهار أكثر من رسم على شاشة واحدة مع امكانية حفظ واسترجاع أى جزء من الشاشة وتتيح كذلك طباعة الرسوم البيانية على طابعات ملونة .

الشركة المنتجة

Pro Works
P.O. Box 1635, Hermiston, OR 97838

dSALVAGE

تحدد الملفات التالفة وتعيدها إلى حالتها الأولى قبل التلف الذى حدث بها وهى تعمل ليس فقط مع ملفات «كلبر» ولكن أيضا مع كل من
dBASE III/III PLUS/IV, Fox Base, dBXL

الشركة المنتجة

COMTECH PUBLISHING LTD, PO BOX 12340, RENO NV 89510

CLIPNET

مكتبة تشتمل على مجموعة من الوظائف لتسهيل التعامل مع شبكة الاتصالات «نـوفـل» وهى معـدة للتطبيقات التى تم تطويرها بقاعدة البيانات «كلبر». وهذه الوظائف مكتوبة بلغة «كلبر» وفي صورة مصدرية ليسهل تعديلها أو توفيقها.

الشركة المنتجة

Data Sync Technologies

P.O. Box 80602 Lansing, MI 48908

ClipOn

مكتبـة تشتمـل على أكثر من ٢٠٠ وظيفة تسهل استخدام الملفات والشاشات والقوائم والنوافذ

الشركة المنتجة

ProWare Corporation

11250–17 Roger Bacon Drive

Reston, Virginia 22090

CLIpx. LIB

مكتبة تشتمل على وظائف تساعد في تطوير نظم ادارة قواعد البيانات. وتخدم هذه الوظائف أغراض معالجة الشاشات والقوائم والنوافذ والجداول.

ZACHARY

مصمم تطبيقـات يساعد في اعداد النظم واستخراج برامج مصدرية وهو مفيد للمبتدئين وقليلى الخبرة بالبرمجة

الشركة المنتجة

Application Programming Incorporated

285 davidson Avenue. somerest, NJ 08873

طبعت بمطابع الابتكار، هاتف ٨٥٧٢٢٠٤ ٠٤/٨٥٧٢٤٠٤ - الدمـــام

هذا الكتاب

يخاطب هذا الكتاب كلا من مبرمجي قاعدة البيانات dBASE III PLUS ومن يرغبون في تطوير نظم لادارة قواعد البيانات باستخدام قاعدة البيانات Clipper والكتاب يشتمل على أربعة أبواب على النحو التالي:

الباب الأول: يشرح مفاهيم أساسية عن تاريخ «كلبر» ومتطلباتها وملفاتها وإمكانياتها وضرورة استخدامها في تطوير النظم والفرق بين المفسر والمترجم ويشرح لمبرمجي dBASE III PLUS كيفية توفيق برامجهم قبل ترجمتها باستخدام «كلبر» ويركز على الامكانيات التي يتميز بها «كلبر» عن «دي بيس» في تطوير البرامج والنظم.

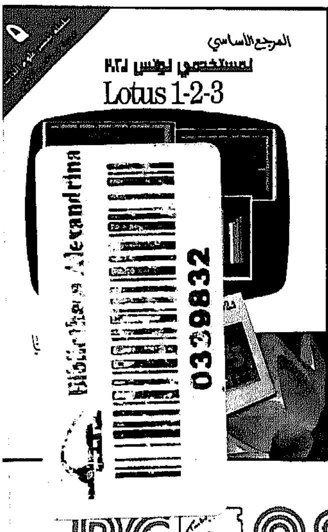
الباب الثاني: يشرح مفاهيم متقدمة تهم بصفة أساسية الذين يرغبون في تطوير أنظمة إدارة قواعد البيانات بإمكانيات متقدمة لا توفرها «دي بيس ثري بلاس» مثل المصفوفات واستخدام قوائم الاختيارات ذات الشريط المضاء والتعامل مع شبكات الاتصالات وكيفية التعامل مع أخطاء البرامج وتعقب واكتشاف الأخطاء.

الباب الثالث: يشرح نظاما متكاملا للمبيعات يشتمل على إجراءات وبرامج حية يمكن استخدامها بصورتها الراهنة أو بعد توفيقها لاعداد نظم إدارة قواعد بيانات مشابهة، والنظام يصلح لخدمة مستفيد واحد أو مجموعة مستفيدين داخل شبكة اتصالات محلية.

الباب الرابع: يشتمل على مرجع شامل لجميع الأوامر والوظائف مرتبة ترتيبا أبجديا لسهولة الوصول إلى أي منها، ويشتمل كل أمر أو وظيفة على معلومات وافية تشمل: شرح مختصر، الشكل العام، الاختيارات المتاحة، الشرح، الاختلاف عن «دي بيس ثري بلاس»، مثال على الأقل، الأوامر والوظائف الأخرى ذات الصلة.







PC NET شبكة الكمبيوتر الشخصي